НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ПОГОДЖЕНО	**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ**
Декан факультету	**Завідувач кафедри комп'ютерних наук**
_____ **Ігор БОЛБОТ**	_____ **Белла ГОЛУБ**
(підпис)	(підпис)
«___» _____ 2026 р.	«___» _____ 2026 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Інформаційна система моніторингу успішності учнів»

Спеціальність «122 Комп'ютерні науки»

Освітньо-професійна програма «Комп'ютерні науки»

Гарант освітньої програми д.е.н., професор	_____ (підпис)	**Роман РУДЕНСЬКИЙ**
Керівник бакалаврської кваліфікаційної роботи к.т.н., доцент	_____ (підпис)	**Ірина БОРОДКІНА**
Виконав	_____ (підпис)	**Даніїл ДОЛОБАШКО**

Київ-2026

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук

к.т.н., доцент _____ Белла ГОЛУБ
(підпис)

«___» _____ 2026 р.

ЗАВДАННЯ
ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
ЗДОБУВАЧУ
Долобашку Данііlу Олександровичу

Спеціальність «122 Комп'ютерні науки»

Освітньо-професійна програма «Комп'ютерні науки»

Тема бакалаврської кваліфікаційної роботи

«Інформаційна система моніторингу успішності учнів»

затверджена наказом від «_26_» ___січня____ 2026 р. № __46___ «С»
Термін подання завершеної роботи на кафедру «___» _____ 2026 р.
Вихідні дані до бакалаврської кваліфікаційної роботи (проєкту): необхідність автоматизації процесу обліку, контролю та аналізу навчальної успішності учнів у закладах загальної середньої освіти
Перелік питань, що підлягають дослідженню:
Системний аналіз предметної області. Інформаційне забезпечення. Прикладне програмне забезпечення. Рекомендації щодо впровадження та експлуатації системи. Висновки.
Перелік графічних документів (за необхідності).
_____

Дата видачі завдання «_6_» _січня_ 2026 р.

Керівник бакалаврської кваліфікаційної роботи _____ (підпис) **Ірина БОРОДКІНА**

Завдання взяв до виконання _____ (підпис) **Данііл ДОЛОБАШКО**

Зміст

Вступ

У сучасних умовах цифровізації освітнього середовища особливої актуальності набуває автоматизація процесів управління навчальною діяльністю учнів та аналізу їхньої успішності. Освітні заклади поступово переходять від паперових журналів і локальних електронних таблиць до повноцінних інформаційних систем, що забезпечують централізоване зберігання, обробку та аналіз навчальних даних. Це дозволяє підвищити ефективність контролю освітнього процесу, спростити взаємодію між учителями, учнями, батьками та адміністрацією школи, а також забезпечити оперативне отримання актуальної інформації щодо успішності учнів.

Актуальність теми бакалаврської кваліфікаційної роботи полягає у необхідності створення сучасної інформаційної системи моніторингу успішності учнів, яка дозволить автоматизувати процеси ведення електронного журналу, контролю оцінювання, аналізу результатів навчання та формування звітності. У багатьох навчальних закладах облік успішності досі здійснюється за допомогою окремих таблиць або паперової документації, що ускладнює обробку інформації, створює ризики втрати даних та потребує значних витрат часу на формування звітів. Відсутність централізованої системи також ускладнює контроль навчального процесу з боку адміністрації та обмежує можливості батьків оперативно отримувати інформацію про результати навчання дітей.

Розробка інформаційної системи моніторингу успішності учнів спрямована на вирішення зазначених проблем шляхом створення єдиного програмного середовища для роботи з навчальними даними. Система забезпечує зручний інтерфейс для внесення оцінок, перегляду статистики успішності, формування звітів та аналізу навчальних показників. Особлива увага приділяється питанням безпеки даних, розмежуванню прав доступу

між різними категоріями користувачів та забезпеченню стабільної роботи програмного забезпечення.

Метою бакалаврської кваліфікаційної роботи є розробка інформаційної системи моніторингу успішності учнів, яка забезпечує автоматизацію процесів обліку та контролю навчальних досягнень, збереження даних про учнів, оцінки, предмети та викладачів, а також надання інструментів для аналізу успішності та формування звітності.

Для досягнення поставленої мети у роботі необхідно вирішити такі завдання:

– провести аналіз предметної області та існуючих інформаційних систем моніторингу успішності учнів;

– визначити функціональні та нефункціональні вимоги до програмного забезпечення;

– виконати моделювання предметної області;

– розробити логічну модель бази даних;

– обґрунтувати вибір системи управління базою даних та інструментів розробки;

– реалізувати програмне забезпечення інформаційної системи;

– провести тестування системи та оцінити результати її роботи;

– сформувати рекомендації щодо впровадження та експлуатації системи.

Об’єктом дослідження є процес моніторингу навчальної успішності учнів у закладах загальної середньої освіти.

Предметом дослідження є методи та засоби розробки інформаційних систем для автоматизації процесів обліку, контролю та аналізу успішності учнів.

У процесі розробки системи використовувалися сучасні підходи та технології програмування, засоби проєктування інформаційних систем і баз даних. Для створення програмного забезпечення застосовувалися інструменти розробки, що забезпечують можливість створення зручного

графічного інтерфейсу користувача, ефективної взаємодії з базою даних та реалізації функцій аналітики успішності.

Практичне значення отриманих результатів полягає у можливості використання розробленої інформаційної системи в закладах освіти для автоматизації процесів ведення обліку успішності учнів, спрощення роботи педагогічного персоналу та покращення контролю навчального процесу. Система може бути адаптована під потреби конкретного навчального закладу та розширена додатковими функціональними можливостями.

Пояснювальна записка бакалаврської кваліфікаційної роботи складається зі вступу, чотирьох основних розділів, висновків, списку використаних джерел та додатків.

У першому розділі проведено системний аналіз предметної області, визначено постановку завдання, здійснено огляд існуючих рішень та виконано моделювання предметної області.

Другий розділ присвячений інформаційному забезпеченню системи, розробці логічної моделі даних, вибору системи управління базою даних та створенню інформаційної бази.

У третьому розділі розглянуто організаційну структуру програмного забезпечення, вибір інструментарію для створення системи, а також алгоритмізацію та програмування основних модулів.

Четвертий розділ містить рекомендації щодо впровадження та експлуатації системи, результати тестування, вимоги до апаратного та програмного забезпечення, а також опис складу інсталяційного пакету.

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Постановка завдання

У сучасних закладах загальної середньої освіти процес обліку та контролю навчальної успішності учнів є одним із ключових елементів організації освітнього процесу. Саме результати навчання дозволяють оцінити рівень засвоєння матеріалу, визначити ефективність методів викладання та своєчасно виявити проблеми у навчанні окремих учнів або цілих класів. Проте в багатьох школах процес моніторингу успішності все ще частково виконується вручну або з використанням окремих електронних таблиць, що створює низку труднощів для педагогічного персоналу та адміністрації.

Традиційні методи ведення журналів успішності мають ряд суттєвих недоліків. Насамперед це значні витрати часу на внесення, пошук та обробку інформації. Учителі змушені вручну підраховувати середні бали, формувати звіти та контролювати динаміку успішності учнів. У випадку паперових журналів існує ризик втрати або пошкодження інформації, а використання великої кількості окремих файлів ускладнює централізоване зберігання даних. Крім того, відсутність автоматизованої системи не дозволяє швидко отримувати аналітичну інформацію щодо результатів навчання та своєчасно реагувати на погіршення успішності.

Однією з основних проблем є складність взаємодії між учителями, учнями, батьками та адміністрацією школи. У більшості випадків інформація про оцінки або відвідування передається із затримкою, а батьки не мають можливості швидко отримувати актуальні дані щодо результатів навчання дитини. Водночас адміністрація навчального закладу змушена

витрачати значний час на збір статистики та підготовку звітної документації.

Для вирішення зазначених проблем доцільним є створення інформаційної системи моніторингу успішності учнів, яка дозволить автоматизувати основні процеси ведення навчальної документації, забезпечити централізоване зберігання даних та надати інструменти для аналізу результатів навчання.

Система повинна забезпечувати роботу з кількома категоріями користувачів, серед яких учителі, учні, батьки та керівники навчального закладу та адміністратор системи. Кожен користувач повинен мати власний рівень доступу до функціональних можливостей системи.

Учитель повинен мати можливість додавати та редагувати оцінки, переглядати інформацію про класи та предмети, формувати статистику успішності й аналізувати результати навчання окремих учнів або класу загалом. Також система повинна забезпечувати автоматичний розрахунок середнього балу та формування звітності.

Учні повинні мати доступ до перегляду власних оцінок, результатів контрольних робіт та загальної статистики успішності. Це дозволить їм самостійно контролювати результати навчання та відстежувати власний прогрес.

Батьки отримують можливість оперативно переглядати інформацію про успішність дитини, середній бал та повідомлення від учителів. Такий підхід дозволяє покращити взаємодію між школою та батьками та забезпечити більш ефективний контроль освітнього процесу.

Керівник навчального закладу або завуч у системі виконує функції контролю, аналізу та організації освітнього процесу. Він має доступ до перегляду статистики успішності учнів, аналізу результатів навчання окремих класів та навчального закладу загалом, формування звітів і контролю роботи педагогічного персоналу. Крім цього, керівник може

створювати оголошення для учнів, батьків та вчителів, додавати інформацію про шкільні події, заходи або зміни в навчальному процесі. Система дозволяє оперативно поширювати важливу інформацію серед користувачів та забезпечує централізоване управління освітнім процесом.

Адміністрація навчального закладу повинна мати розширені можливості управління системою. До основних функцій адміністратора належать створення облікових записів користувачів та редагування інформації про класи.

Одним із ключових елементів системи повинен бути модуль електронного журналу та роботи з оцінками. Система повинна забезпечувати можливість перегляду списку учнів класу, додавання та редагування оцінок з навчальних предметів, а також автоматичного збереження інформації у базі даних. Крім цього, система повинна забезпечувати швидкий доступ до результатів навчання учнів та підтримувати формування статистики успішності у вигляді таблиць і графічного представлення даних для аналізу результатів навчального процесу.

Приклад візуалізації статистики успішності наведено на рисунку 1.1.

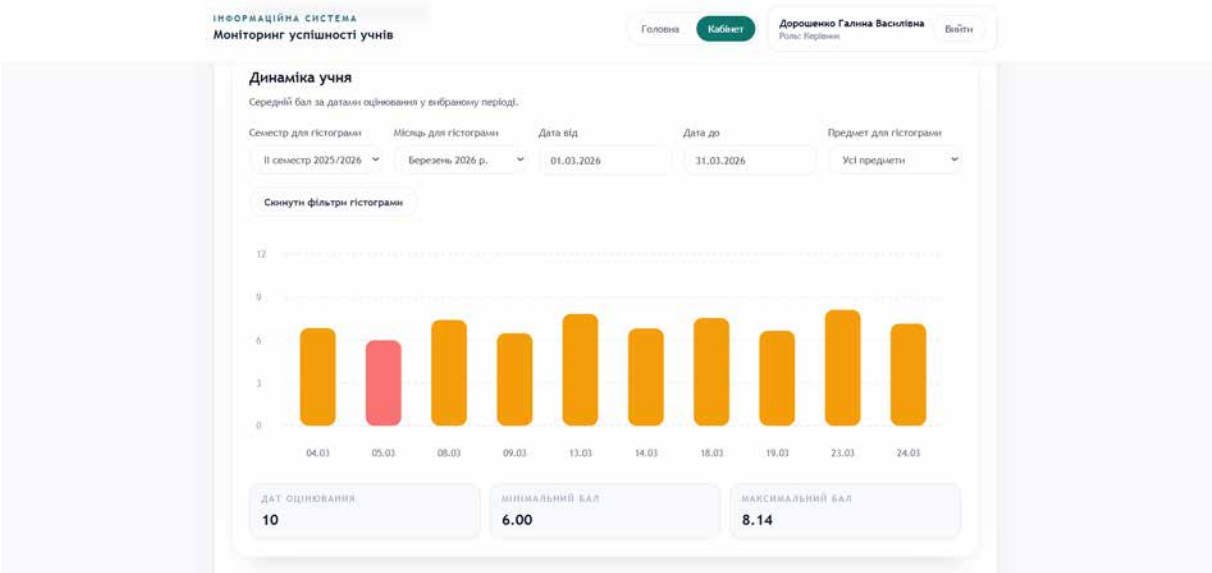


Рис. 1.1 Графічне представлення оцінок учня за період

Важливою вимогою до системи є забезпечення безпеки даних. Оскільки система працює з персональною інформацією учнів та працівників навчального закладу, необхідно реалізувати механізми автентифікації та авторизації користувачів. Кожен користувач повинен мати доступ лише до тих функцій і даних, які відповідають його ролі у системі.

Інтерфейс системи повинен бути інтуїтивно зрозумілим та зручним у використанні. Особливу увагу необхідно приділити логічній структурі меню та простоті навігації. Оскільки системою користуватимуться люди з різним рівнем цифрової грамотності, важливо забезпечити максимально простий механізм взаємодії з програмним забезпеченням.

Приклад головного меню системи наведено на рисунку 1.2.

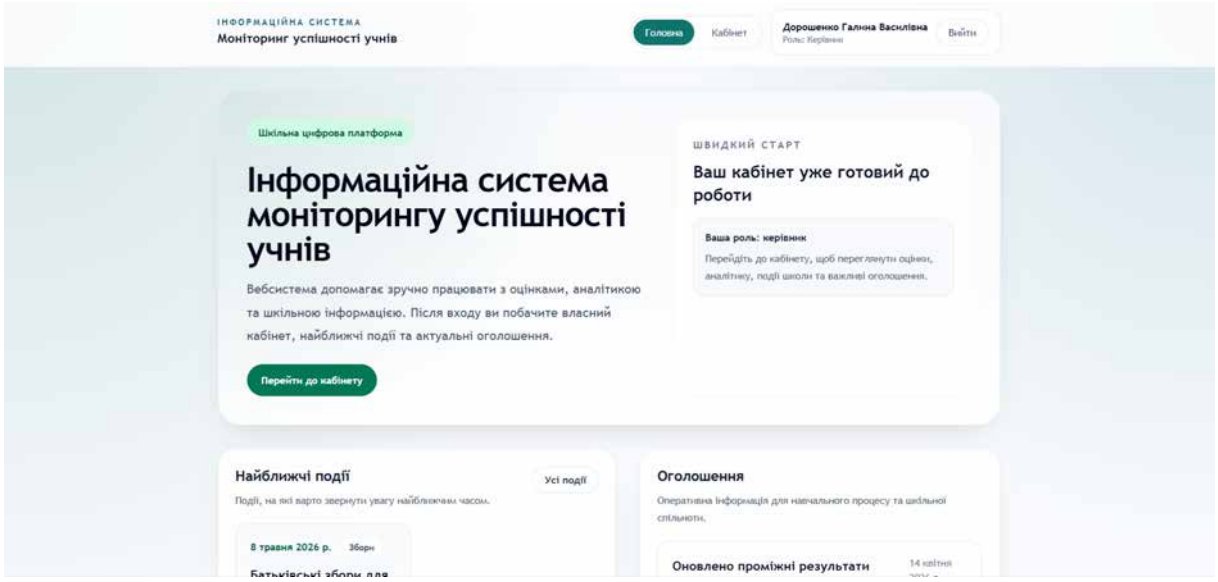


Рис. 1.2 Головне меню

До нефункціональних вимог системи належать стабільність роботи, висока продуктивність, можливість масштабування та сумісність із сучасними операційними системами. Система повинна коректно працювати навіть при великій кількості користувачів та забезпечувати швидке виконання основних операцій.

Таким чином, розробка інформаційної системи моніторингу успішності учнів є актуальним завданням, спрямованим на підвищення

ефективності організації освітнього процесу шляхом автоматизації обліку, контролю та аналізу результатів навчання.

1.2 Огляд інформаційних джерел та існуючих рішень

Сучасний розвиток інформаційних технологій сприяв активному впровадженню електронних систем управління навчальним процесом у закладах освіти. Використання цифрових платформ дозволяє автоматизувати процеси ведення документації, спростити взаємодію між учасниками освітнього процесу та забезпечити швидкий доступ до інформації про результати навчання.

Одним із найбільш поширених напрямків автоматизації освітнього процесу є використання електронних журналів та систем дистанційного навчання. Такі системи забезпечують можливість ведення обліку оцінок, контролю відвідування, формування статистики та організації дистанційної взаємодії між учителями й учнями.

На сьогодні існує велика кількість програмних рішень для автоматизації освітнього процесу. Найбільш популярними серед них є Google Classroom, Moodle, «Єдина школа» та Human.

Google Classroom є однією з найпоширеніших платформ для організації дистанційного навчання. Система дозволяє створювати навчальні курси, публікувати завдання, оцінювати роботи учнів та організовувати обмін файлами між учасниками навчального процесу.

Інтерфейс системи Google Classroom наведено на рисунку 1.3.

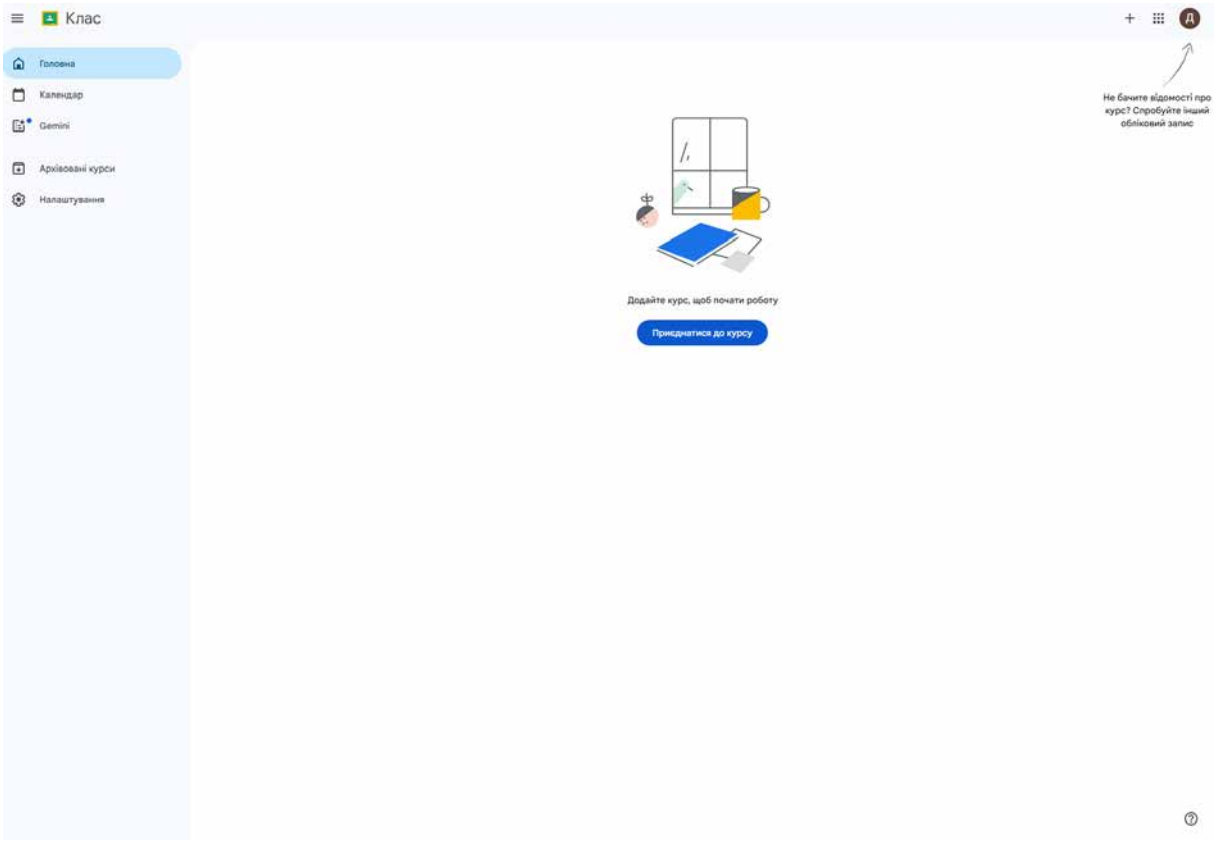


Рис 1.3 Інтерфейс системи Google Classroom

Основною перевагою Google Classroom є інтеграція з іншими сервісами Google, що значно спрощує організацію навчального процесу. Проте система орієнтована переважно на дистанційне навчання та не забезпечує широких можливостей для аналізу успішності учнів.

Ще однією популярною платформою є Moodle. Це система дистанційного навчання з відкритим вихідним кодом, яка активно використовується у навчальних закладах різних країн світу.

Приклад інтерфейсу системи Moodle наведено на рисунку 1.4.

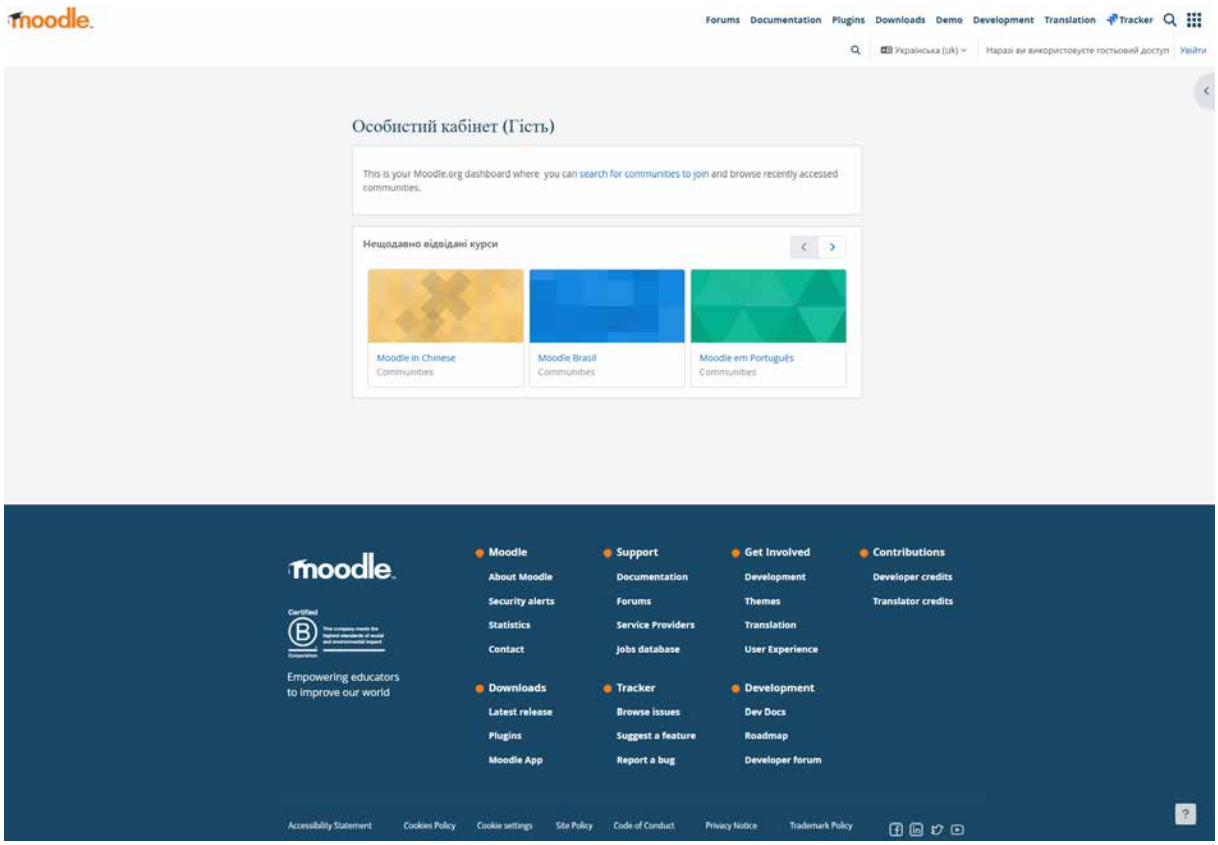


Рис. 1.4 Інтерфейс системи Moodle

Система Moodle забезпечує створення електронних курсів, тестування, ведення журналів оцінок та формування звітності. Основною перевагою системи є широкі можливості налаштування та масштабування. Водночас через значну кількість функцій система може бути складною для використання у звичайних закладах середньої освіти.

В Україні значного поширення набули системи «Єдина школа» та Human, які орієнтовані безпосередньо на автоматизацію навчального процесу в закладах загальної середньої освіти.

Система «Єдина школа» забезпечує функції електронного журналу, електронного щоденника, розкладу занять та формування звітності.

Інтерфейс системи «Єдина школа» наведено на рисунку 1.5.

Рис. 1.5 Інтерфейс системи «Єдина школа»

Платформа Human також забезпечує можливість ведення електронного журналу, створення домашніх завдань, тестування та комунікації між учителями, учнями та батьками.

Приклад інтерфейсу системи Human наведено на рисунку 1.6.

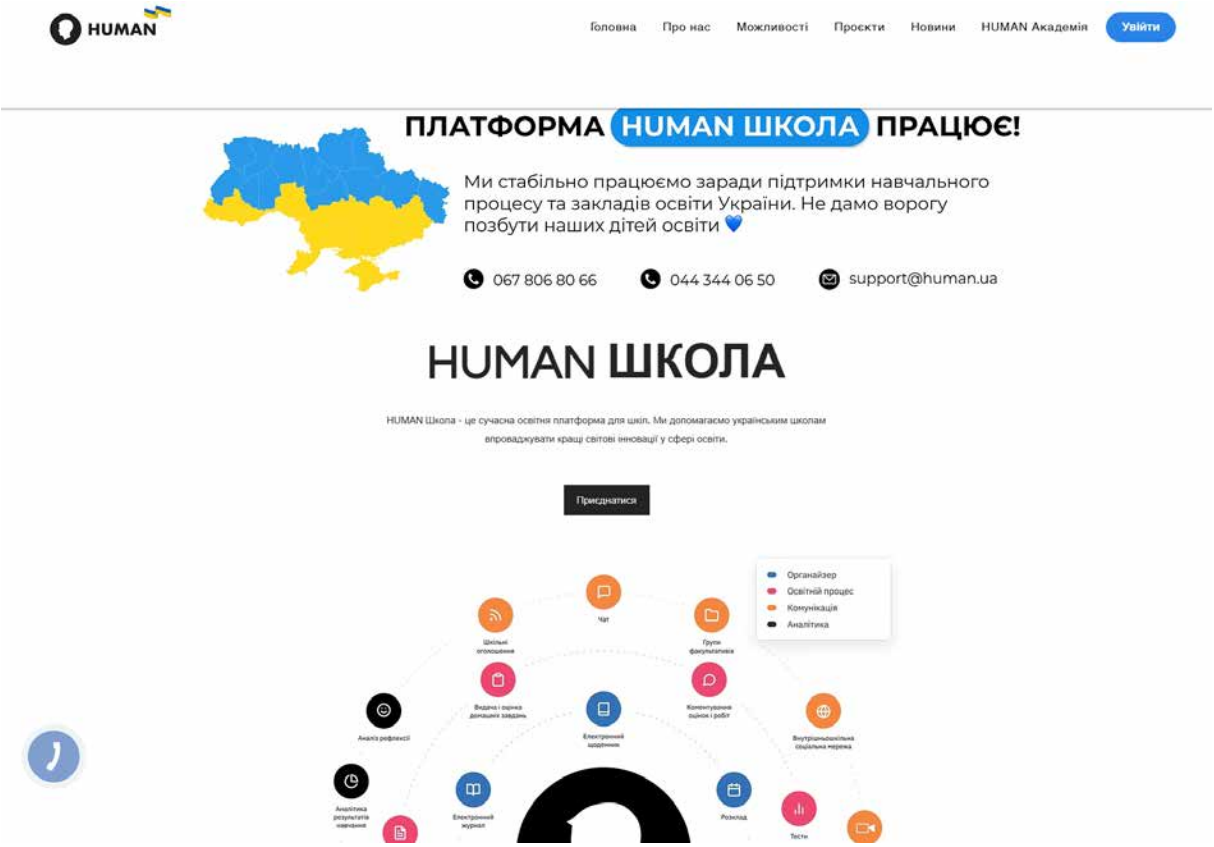


Рис. 1.6 Інтерфейс системи Human

Аналіз існуючих систем показує, що більшість сучасних платформ мають широкий функціонал та забезпечують автоматизацію основних процесів організації навчання. Проте значна частина таких систем характеризується складним інтерфейсом, високими вимогами до технічних ресурсів або надлишковою функціональністю.

Під час аналізу було встановлено, що ефективна система моніторингу успішності повинна поєднувати простий інтерфейс користувача, високу швидкість роботи, можливість централізованого зберігання даних та наявність аналітичних інструментів для оцінки результатів навчання.

Особливу увагу в сучасних інформаційних системах приділяють візуалізації статистичних даних. Побудова графіків і діаграм дозволяє швидко аналізувати результати навчання окремих учнів або класів загалом.

Приклад статистичної аналітики успішності наведено на рисунку 1.7.

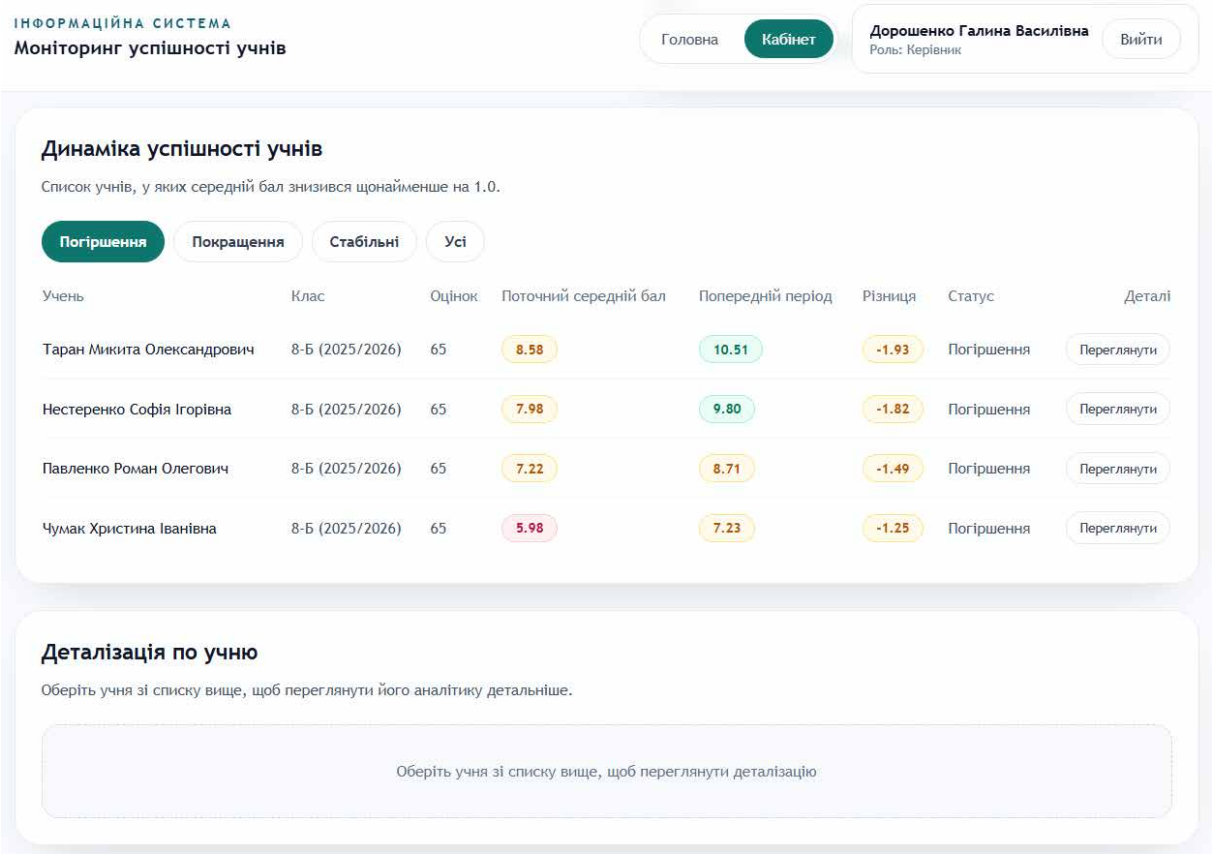


Рис. 1.7 Блок аналітики успішності учнів

Ще одним важливим аспектом сучасних освітніх систем є забезпечення інформаційної безпеки. Оскільки системи працюють із персональними даними учнів та працівників навчального закладу, необхідно реалізувати механізми захисту інформації від несанкціонованого доступу.

Таким чином, аналіз існуючих рішень показує, що сучасні системи моніторингу успішності забезпечують широкий набір функцій для автоматизації освітнього процесу, проте багато з них є складними або надлишковими для використання у звичайних закладах загальної середньої освіти. Саме тому доцільною є розробка власної інформаційної системи, орієнтованої на простоту використання, зручний інтерфейс та ефективний контроль результатів навчання учнів.

1.3 Моделювання предметної області

Після проведення аналізу предметної області та дослідження існуючих інформаційних систем наступним етапом розробки є моделювання предметної області. Даний етап дозволяє сформувати структуру майбутньої інформаційної системи, визначити основні компоненти, взаємозв’язки між ними та описати логіку функціонування програмного забезпечення.

Моделювання предметної області є важливим етапом проєктування, оскільки саме на цьому етапі формується загальна архітектура системи та визначаються основні бізнес-процеси. Крім того, побудова моделей дозволяє значно спростити подальшу реалізацію програмного забезпечення та зменшити кількість помилок під час програмування.

Для моделювання інформаційної системи моніторингу успішності учнів доцільно використовувати UML-діаграми, які дозволяють наочно представити структуру системи, поведінку користувачів та взаємозв’язки між основними об’єктами.

На початковому етапі моделювання було визначено основних учасників системи. До них належать учитель, учень, батьки, керівник навчального закладу або завуч, а також адміністратор системи. Кожен користувач взаємодіє із системою відповідно до власних функціональних можливостей та рівня доступу до інформації.

Загальна взаємодія користувачів із системою наведена на діаграмі прецедентів на рисунку 1.8.

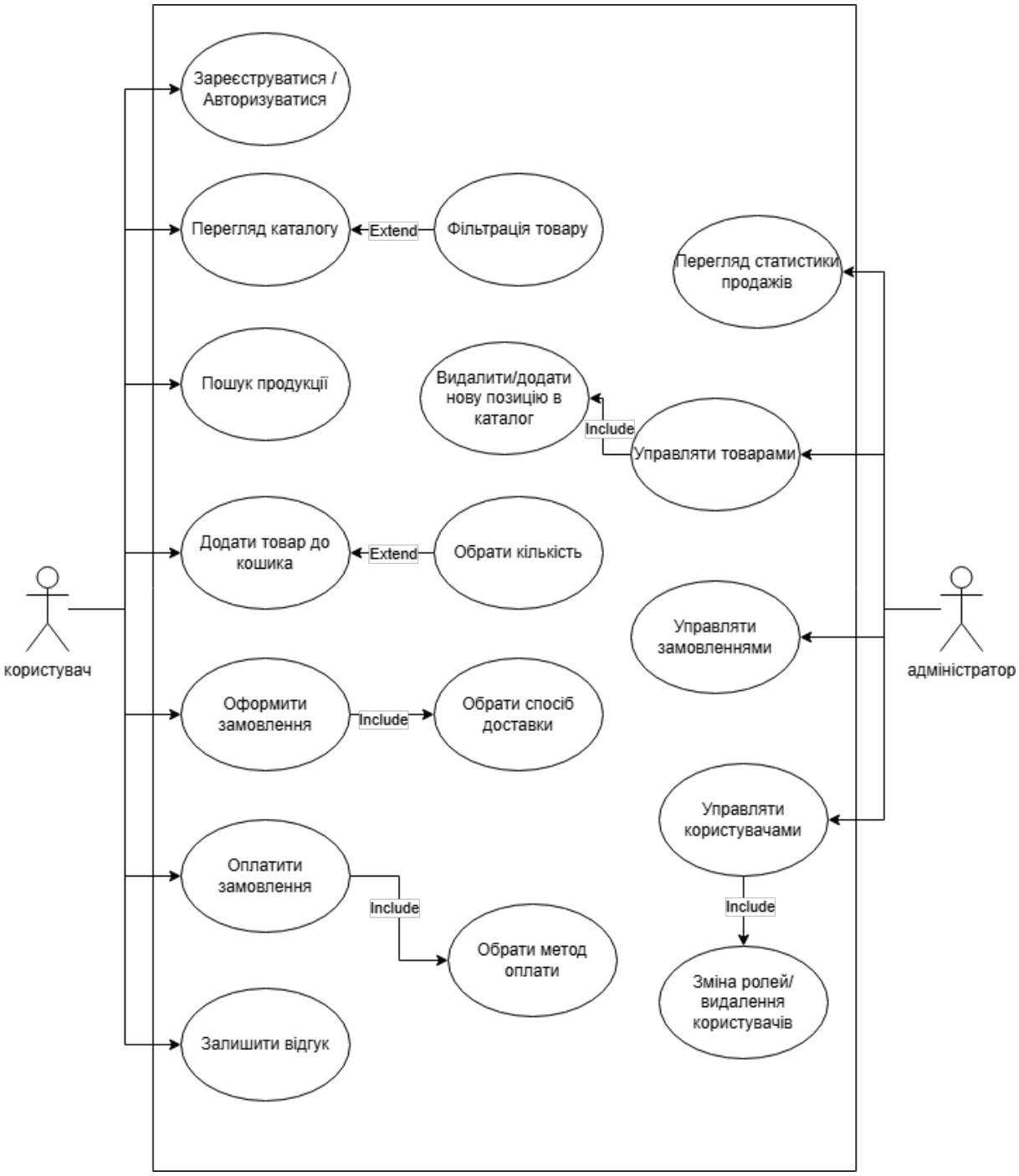


Рис. 1.8 Діаграма прецендентів

Учитель є одним із ключових користувачів системи, оскільки саме він виконує основну роботу з ведення електронного журналу та контролю успішності учнів. Учитель має можливість додавати та редагувати оцінки, переглядати інформацію про учнів, формувати статистику успішності та створювати звіти.

Учень взаємодіє із системою переважно в режимі перегляду інформації. Він має можливість переглядати власні оцінки, середній бал, результати контрольних робіт та аналізувати власну успішність за певний період часу.

Батьки отримують доступ до інформації про результати навчання дитини, що дозволяє оперативно контролювати зміни успішності та взаємодіяти з учителями у разі виникнення проблем у навчанні.

Керівник навчального закладу або завуч виконує функції аналізу та контролю освітнього процесу. Він має можливість переглядати статистику успішності класів, аналізувати результати навчання, формувати загальні звіти та контролювати ефективність роботи педагогічного персоналу.

Адміністратор системи відповідає за підтримку працездатності програмного забезпечення, управління користувачами та забезпечення інформаційної безпеки. До його функцій належать створення облікових записів, налаштування ролей користувачів, керування класами та контроль коректної роботи системи.

Одним із головних процесів у системі є внесення оцінок до електронного журналу. Для більш детального опису цього процесу було побудовано діаграму діяльності.

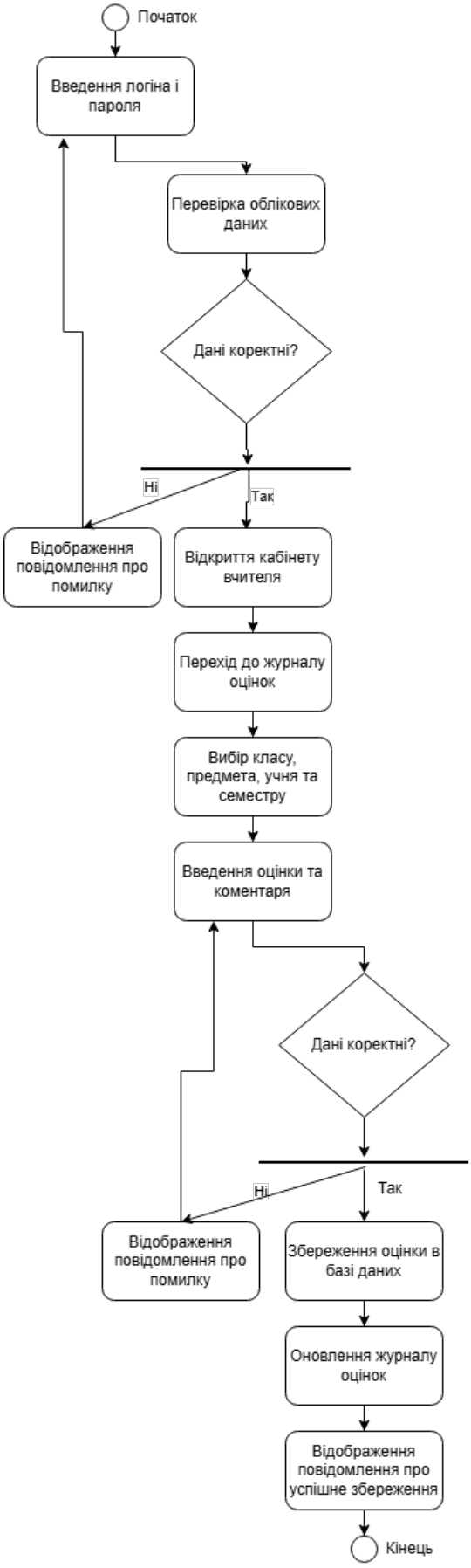


Рис. 1.9 Діаграма діяльності внесення оцінок

Процес починається з авторизації користувача у системі. Після успішного входу вчитель отримує доступ до електронного журналу та обирає необхідний клас і навчальний предмет. Далі виконується вибір учня та введення оцінки. Після підтвердження введених даних система автоматично зберігає інформацію у базі даних та оновлює статистику успішності.

Ще одним важливим процесом є формування звітів щодо результатів навчання. Інформаційна система повинна забезпечувати можливість створення звітності для окремих учнів, класів або навчальних предметів.

Після авторизації користувач обирає тип звіту та необхідний період формування статистики. Система виконує обробку інформації, формує звіт та відображає його у вигляді таблиці або графічної статистики. За необхідності користувач може експортувати звіт у файл.

Під час моделювання також було визначено основні функціональні модулі системи. Інформаційна система повинна складатися з:

– модуля авторизації;
– модуля управління користувачами;
– модуля електронного журналу;
– модуля статистики та аналітики;
– модуля формування звітів;
– модуля адміністрування.

Загальна структура програмних модулів наведена на схемі архітектури системи.

Модуль авторизації відповідає за перевірку облікових даних користувачів та забезпечення безпечного доступу до системи. Для кожної категорії користувачів встановлюється окремий рівень доступу до функціональних можливостей.

Модуль електронного журналу забезпечує додавання, редагування та перегляд оцінок. Також система автоматично обчислює середній бал учня та формує статистику успішності.

Модуль статистики та аналітики відповідає за обробку інформації та графічне представлення результатів навчання. Це дозволяє швидко аналізувати успішність окремих учнів, класів або навчальних предметів.

Модуль формування звітів дозволяє автоматично створювати документи щодо успішності учнів за певний період часу. Звіти можуть формуватися у вигляді таблиць, графіків або Excel таблиць.

Особливу увагу під час моделювання системи було приділено питанням інформаційної безпеки. Оскільки система працює з персональними даними учнів та працівників навчального закладу, необхідно реалізувати механізми захисту інформації від несанкціонованого доступу.

Для забезпечення безпеки даних система повинна підтримувати автентифікацію користувачів, розмежування прав доступу та захист облікових даних.

Після введення логіна та пароля система виконує перевірку облікових даних користувача. У разі успішної авторизації користувач отримує доступ до функціональних можливостей відповідно до своєї ролі. Якщо дані введено неправильно, система відображає повідомлення про помилку.

У процесі моделювання також було визначено основні нефункціональні вимоги до програмного забезпечення. Система повинна забезпечувати стабільну роботу, швидке виконання операцій, зручний інтерфейс користувача та можливість подальшого масштабування.

Інтерфейс користувача повинен бути максимально простим та інтуїтивно зрозумілим для користувачів різного рівня підготовки. Особливу увагу необхідно приділити логічному розташуванню елементів меню,

швидкому доступу до основних функцій та зручності роботи з електронним журналом.

Таким чином, моделювання предметної області дозволило визначити основні компоненти інформаційної системи, функціональні модулі та взаємозв’язки між основними об’єктами. Отримані результати будуть використані на наступних етапах проєктування бази даних та реалізації програмного забезпечення.

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Проєктування логічної моделі даних

Одним із найважливіших етапів розробки інформаційної системи моніторингу успішності учнів є проєктування логічної моделі даних. Саме від правильно побудованої структури бази даних залежить стабільність роботи програмного забезпечення, швидкість доступу до інформації, зручність формування аналітики та можливість подальшого розширення функціоналу системи. Логічна модель дає змогу визначити основні сутності предметної області, їх атрибути та взаємозв'язки між ними. Інформаційна система моніторингу успішності учнів забезпечує централізоване зберігання даних, пов'язаних із навчальним процесом. До таких даних належать відомості про користувачів системи, їхні ролі, учнів, учителів, батьків, керівництво навчального закладу, навчальні класи, предмети, семестри, оцінки, призначення вчителів до предметів і класів, а також шкільні події. Уся інформація організована таким чином, щоб забезпечити швидкий доступ до потрібних даних, зручну фільтрацію, пошук та формування аналітичних звітів.

Під час побудови логічної моделі було визначено основні сутності інформаційної системи та встановлено зв'язки між ними. До ключових об'єктів системи належать: користувачі, ролі, учні, учителі, батьки, керівники навчального закладу, класи, навчальні предмети, семестри, оцінки, зв'язки між батьками та учнями, а також зв'язки між учителями, предметами і класами.

Сутність **«Користувач»** є базовою для організації доступу до системи. У ній зберігаються загальні облікові дані, необхідні для входу до системи, зокрема ім'я, прізвище, по батькові, електронна пошта, хеш пароля, статус активності облікового запису, роль користувача та дата

створення. Окрема таблиця користувачів дозволяє не дублювати ці дані в профілях учнів, учителів, батьків і керівників.

Сутність **«Роль»** визначає рівень доступу користувача до функцій інформаційної системи. У системі передбачено ролі учня, батьків, учителя, керівника та адміністратора. Адміністратор у даній реалізації не є окремою таблицею бази даних, а реалізований як звичайний користувач із відповідною роллю доступу.

Сутність **«Учень»** є однією з центральних у системі, оскільки саме навколо неї формується основна інформація про результати навчання. Для кожного учня система зберігає зв’язок із користувачем, прив’язку до класу, дату народження та службові примітки. Інформація про успішність учня не зберігається у вигляді окремого поля середнього балу, а обчислюється динамічно на основі таблиці оцінок.

Сутність **«Учитель»** містить інформацію про педагогічного працівника та пов’язана з відповідним обліковим записом користувача. Відомості про те, які саме предмети та в яких класах викладає вчитель, зберігаються не безпосередньо в таблиці вчителів, а через окрему таблицю призначень учителів.

Сутність **«Батьки»** використовується для представлення законних представників учнів у системі. Вона також пов’язана з обліковим записом користувача. Оскільки один із батьків може бути пов’язаний з кількома дітьми, а один учень може мати декількох законних представників, для реалізації такого зв’язку використовується окрема проміжна таблиця.

Сутність **«Керівник»** використовується для представлення адміністрації закладу освіти або завуча як окремого профілю, пов’язаного з обліковим записом користувача. Користувач із цією роллю має доступ до аналітики успішності, порівняння результатів по класах, предметах і учнях, а також до формування звітів.

Сутність **«Клас»** використовується для об’єднання учнів у навчальні групи. Для кожного класу зберігається його назва та навчальний рік. Клас пов’язаний з учнями та з призначеннями вчителів до предметів. Це дозволяє організувати структуру навчального процесу за класами й навчальними роками.

Сутність **«Предмет»** містить інформацію про навчальні дисципліни, що викладаються у школі. Для кожного предмета зберігається назва та, за потреби, короткий опис. Зв’язок предмета з конкретним учителем і класом реалізується через таблицю призначень учителів.

Сутність **«Семестр»** використовується для поділу навчального року на часові періоди. Для кожного семестру зберігаються його назва, дата початку, дата завершення та ознака поточного семестру. Це дозволяє реалізовувати аналітику успішності у розрізі окремих навчальних періодів.

Сутність **«Оцінка»** є основним елементом моніторингу успішності. Вона містить інформацію про учня, учителя, предмет, семестр, значення оцінки, дату виставлення, коментар до оцінки та дату створення запису. Саме на основі таблиці оцінок формуються середній бал, аналітика успішності, порівняння за періодами та звітність.

Сутність **«Призначення вчителя»** використовується для збереження інформації про те, який учитель викладає який предмет у якому класі. Ця таблиця є важливою для логіки навчального процесу, оскільки дозволяє реалізувати гнучку модель викладання, коли один учитель може працювати з кількома класами, а в одному класі викладається багато предметів.

Сутність **«Зв’язок батьків та учнів»** використовується для організації зв’язку типу багато-до-багатьох між батьками та дітьми. Завдяки цій таблиці один із батьків може бути пов’язаний з кількома учнями, а один учень — з одним або кількома законними представниками.

Сутність **«Шкільні події»** використовується для збереження інформації про оголошення, заходи, зустрічі та інші події навчального

закладу. Ця частина системи розширює функціональні можливості платформи та забезпечує додаткову інформаційну взаємодію між учасниками освітнього процесу. Взаємозв’язки між основними сутностями наведено на ER-діаграмі.

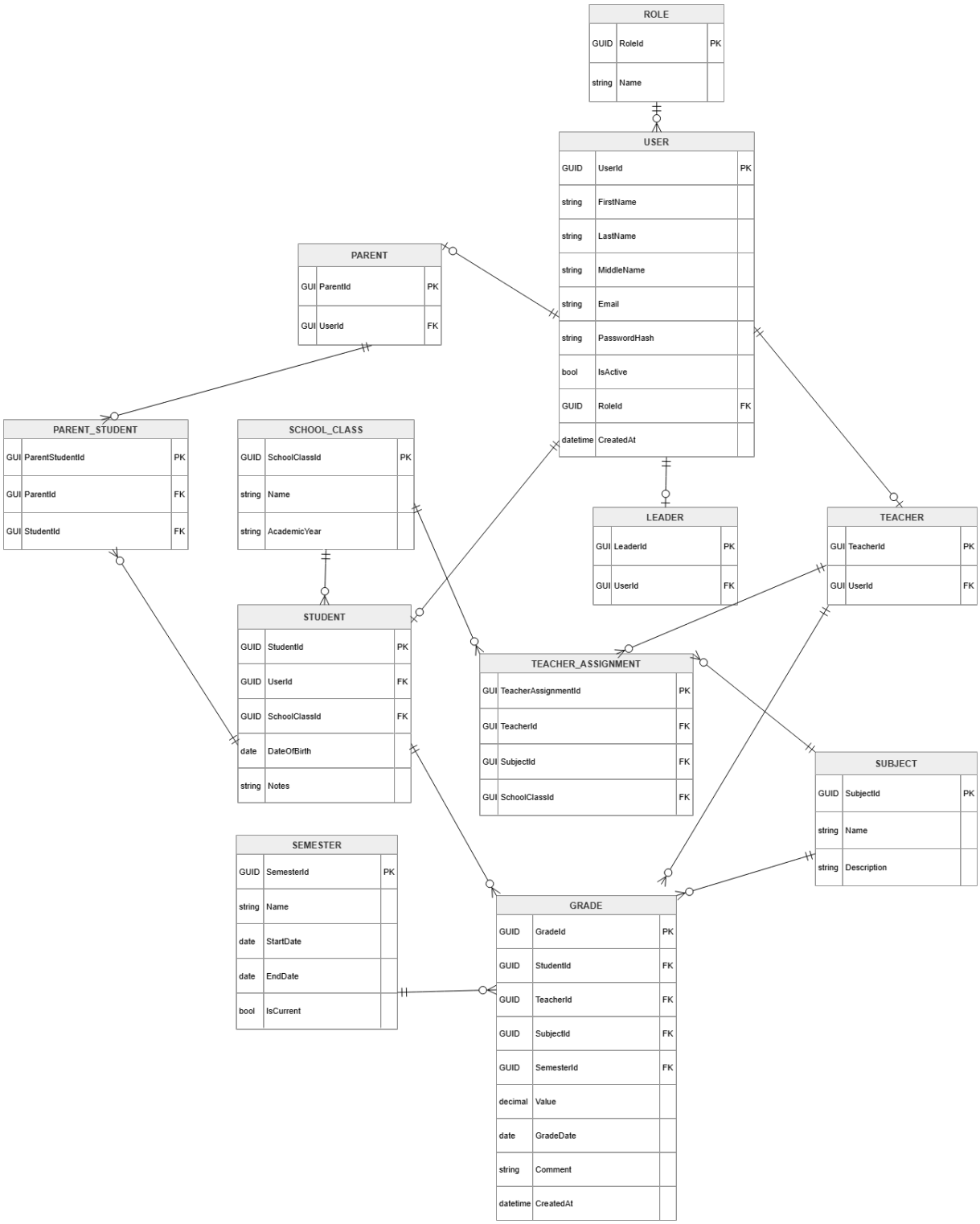


Рис. 2.1 ER-діаграма сутностей логічної моделі

ER-діаграма дозволяє визначити логічну структуру бази даних і описати зв’язки між основними сутностями системи. Один клас містить багато учнів, а один учень може мати багато оцінок із різних предметів у межах різних семестрів. Один учитель може викладати декілька дисциплін і працювати з кількома класами. Батьки можуть бути пов’язані з одним або кількома учнями. Керівник навчального закладу має доступ до загальної аналітики успішності, а адміністратор забезпечує керування користувачами та технічне супроводження системи через механізм ролей доступу.

Під час проєктування логічної моделі особливу увагу було приділено нормалізації бази даних. Це дозволило зменшити надлишковість інформації, уникнути дублювання даних та спростити процес оновлення записів. Структура таблиць побудована з урахуванням принципів нормалізації, що забезпечує ефективне зберігання інформації та підвищує продуктивність роботи системи.

Одним із важливих аспектів інформаційного забезпечення є організація пошуку та фільтрації даних. Оскільки система працює з великою кількістю записів, користувачі повинні мати можливість швидко знаходити необхідну інформацію про учнів, оцінки, класи або результати навчання. У системі реалізовано локальний текстовий пошук та механізми фільтрації за ролями, класами, предметами, семестрами, датами та іншими параметрами.

Для забезпечення швидкої роботи системи використовуються механізми індексації таблиць бази даних. Індекси дозволяють скоротити час виконання запитів і підвищити продуктивність бази даних під час роботи з великим обсягом інформації. Особливо важливими є індекси для таблиць користувачів, учнів, оцінок, класів і зв’язків між сутностями.

Під час проєктування логічної моделі також було враховано можливість подальшого масштабування системи. У майбутньому програмне забезпечення може бути розширене додатковими модулями,

наприклад для контролю відвідуваності, роботи з домашніми завданнями, організації дистанційного навчання або створення мобільного застосунку.

Таким чином, побудована логічна модель даних дозволяє забезпечити ефективне зберігання інформації, швидкий доступ до даних та стабільну роботу інформаційної системи моніторингу успішності учнів. Отримані результати були використані на наступних етапах проєктування та реалізації бази даних.

2.2 Вибір системи управління інформаційною базою

Після побудови логічної моделі даних наступним етапом розробки інформаційної системи є вибір системи управління базою даних. Саме система управління базою даних забезпечує зберігання, обробку, зміну та захист інформації, яка використовується під час роботи програмного забезпечення.

Вибір СУБД є одним із найважливіших етапів проєктування інформаційної системи, оскільки від цього залежить продуктивність роботи системи, стабільність зберігання інформації, швидкість виконання запитів та можливість подальшого масштабування програмного забезпечення. Для системи моніторингу успішності учнів особливо важливо забезпечити швидкий доступ до даних, підтримку великої кількості записів та надійний захист персональної інформації.

На сучасному етапі існує велика кількість систем управління базами даних, які відрізняються архітектурою, функціональними можливостями та сферою застосування. Найбільш поширеними реляційними системами управління базами даних є MySQL, PostgreSQL, Oracle Database та Microsoft SQL Server.

Однією з поширених СУБД є MySQL. Вона активно використовується у веброзробці, має прості механізми налаштування та забезпечує достатньо

високу швидкість роботи при вирішенні широкого кола прикладних задач. Іншою популярною системою є PostgreSQL. Вона характеризується високим рівнем надійності, підтримкою складних SQL-запитів та розширеними можливостями роботи з аналітичними запитами й великими обсягами інформації.

Також однією з найпотужніших СУБД є Oracle Database. Вона використовується переважно у великих корпоративних проєктах і забезпечує високу продуктивність, надійність та масштабованість. Водночас її застосування доцільне насамперед у великих системах через складність адміністрування та високі вимоги до ресурсів.

Для реалізації інформаційної системи моніторингу успішності учнів було обрано **Microsoft SQL Server**. Дана система управління базами даних є сучасною реляційною СУБД, яка забезпечує високу швидкість роботи, стабільність, підтримку складних зв’язків між таблицями, зручні засоби адміністрування та хорошу сумісність із технологіями, використаними під час розробки програмного забезпечення.

Серед основних причин вибору Microsoft SQL Server можна виділити: простоту інтеграції з програмним забезпеченням, створеним на платформі .NET, підтримку мови SQL, високу продуктивність, наявність зручного середовища адміністрування SQL Server Management Studio, підтримку механізмів цілісності даних, можливість масштабування структури бази даних у майбутньому.

Важливою перевагою Microsoft SQL Server є підтримка реляційної моделі даних, що дозволяє ефективно працювати зі складними взаємозв’язками між сутностями. Оскільки система моніторингу успішності містить значну кількість взаємопов’язаних таблиць, використання реляційної СУБД є найбільш доцільним рішенням.
Ще однією перевагою Microsoft SQL Server є підтримка механізмів забезпечення цілісності даних. За допомогою первинних та зовнішніх

ключів система контролює правильність зв’язків між таблицями та запобігає появі помилкових або непов’язаних записів у базі даних.

Для роботи інформаційної системи особливо важливо забезпечити швидке виконання запитів. Microsoft SQL Server підтримує механізми індексації, які дозволяють значно підвищити продуктивність під час пошуку, фільтрації та обробки інформації.

Індекси використовуються для прискорення роботи з таблицями, які містять інформацію про користувачів, учнів, оцінки, класи та інші об’єкти системи. Завдяки цьому користувачі можуть швидко отримувати потрібну інформацію навіть при великій кількості записів.

Розмежування прав доступу в інформаційній системі реалізовано на рівні прикладної логіки програмного забезпечення із використанням ролей користувачів, механізмів автентифікації та авторизації. Це дозволяє забезпечити різні рівні доступу до функцій системи для вчителів, учнів, батьків, керівництва навчального закладу та адміністратора.

Наприклад, учитель має можливість додавати та редагувати оцінки, але не може змінювати системні налаштування. Учні та батьки мають доступ переважно до перегляду інформації, а адміністратор має можливість керувати користувачами та підтримувати працездатність системи. Окрему увагу було приділено питанню масштабованості. Microsoft SQL Server дозволяє поступово розширювати структуру бази даних та додавати нові функціональні модулі без необхідності повного перепроєктування всієї інформаційної бази. У майбутньому система може бути доповнена новими підсистемами, зокрема модулями відвідуваності, дистанційного навчання, домашніх завдань або мобільним застосунком.

Під час вибору СУБД також враховувалися питання сумісності з мовами програмування та інструментами розробки. Microsoft SQL Server добре інтегрується з платформою .NET, а для роботи прикладної частини з базою даних у проєкті використовуються сучасні засоби доступу до даних.

Для взаємодії програмного забезпечення з базою даних у системі використовується мова SQL, а основна робота з інформацією реалізована за допомогою технології Entity Framework Core. Для окремих технічних операцій масового запису даних також можуть застосовуватися низькорівневі механізми доступу до SQL Server.

Таким чином, у результаті аналізу сучасних систем управління базами даних для реалізації інформаційної системи моніторингу успішності учнів було обрано Microsoft SQL Server. Дана СУБД забезпечує високу продуктивність, стабільність роботи, захист інформації та можливість подальшого розвитку програмного забезпечення. Її використання дозволяє створити надійну інформаційну базу для ефективної роботи системи моніторингу успішності учнів.

2.3 Створення інформаційної бази

Після завершення етапу проєктування логічної моделі даних та вибору системи управління базою даних наступним етапом розробки стало створення інформаційної бази. На даному етапі виконувалася фізична реалізація структури бази даних у середовищі Microsoft SQL Server, створення таблиць, налаштування взаємозв’язків між ними та реалізація механізмів роботи з інформацією.

Інформаційна база є основою всієї інформаційної системи, оскільки саме вона забезпечує зберігання, обробку та швидкий доступ до даних. Від правильності побудови структури бази даних залежить стабільність роботи системи, продуктивність виконання запитів та можливість подальшого розвитку програмного забезпечення.

На початковому етапі у середовищі SQL Server Management Studio було створено окрему базу даних для інформаційної системи моніторингу

успішності учнів. Після цього було реалізовано структуру таблиць відповідно до побудованої логічної моделі.

До складу інформаційної бази увійшли таблиці для зберігання інформації про користувачів, ролі, учнів, учителів, батьків, керівників, класи, предмети, семестри, оцінки, шкільні події, а також таблиці зв'язків між батьками та учнями і між учителями, предметами та класами. Кожна таблиця містить набір полів відповідно до функціонального призначення сутності.

Таблиця **«Users»** використовується для зберігання облікових даних користувачів системи. Таблиця **«Roles»** визначає ролі доступу користувачів.

Таблиці **«Students»**, **«Teachers»**, **«Parents»** та **«Leaders»** містять профілі відповідних категорій користувачів і пов'язані з таблицею користувачів. Таблиця **«SchoolClasses»** використовується для зберігання інформації про навчальні класи та навчальні роки. Таблиця **«Subjects»** містить перелік навчальних предметів. Таблиця **«Semesters»** використовується для опису часової структури навчального процесу.

Однією з ключових таблиць є таблиця **«Grades»**, оскільки саме вона використовується для зберігання результатів навчальної діяльності учнів. У ній зберігаються оцінка, дата виставлення, коментар, а також зв'язки з учнем, учителем, предметом і семестром.

Для реалізації зв'язків між учителями, класами та предметами створено таблицю **«TeacherAssignments»**. Вона забезпечує правильне відображення того, який учитель викладає певний предмет у конкретному класі. Для реалізації зв'язків між батьками та дітьми використовується таблиця **«ParentStudents»**.

Окремо в системі передбачено таблицю **«SchoolEvents»**, яка зберігає інформацію про події навчального закладу. Це дозволяє використовувати

інформаційну систему не лише для моніторингу оцінок, а й для інформаційного супроводу шкільного життя.

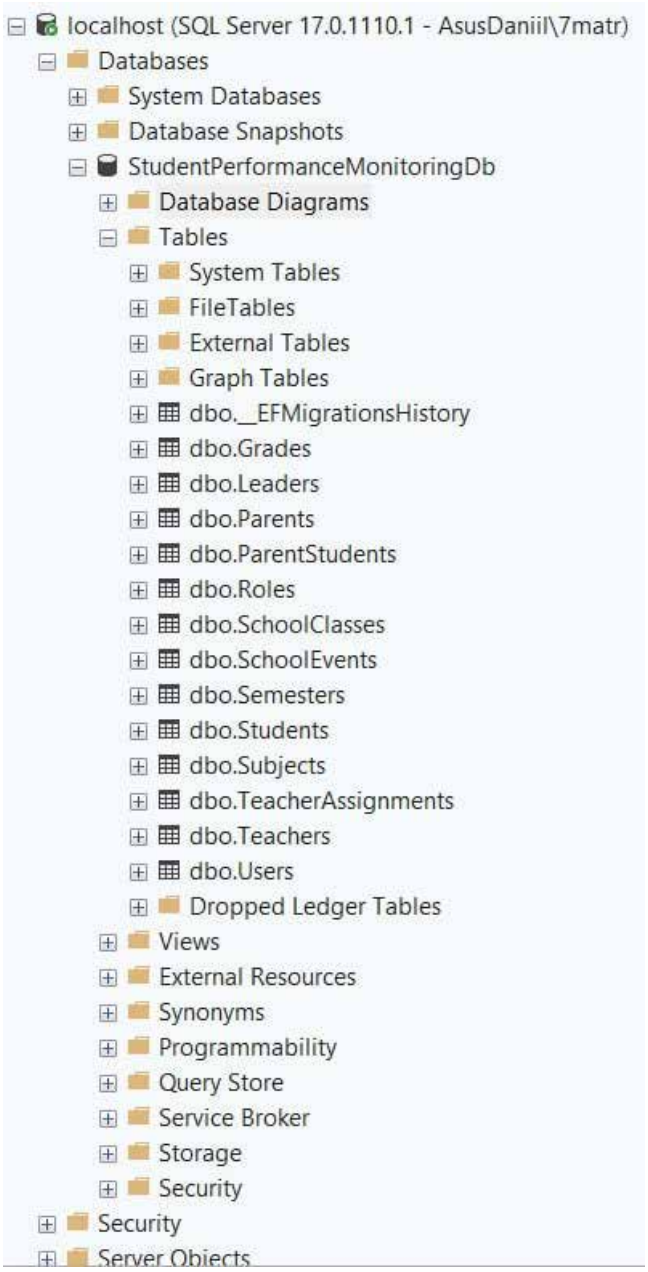


Рис. 2.2 Структура бази даних у SQL Server Management Studio

Після створення таблиць було реалізовано взаємозв’язки між ними. Для забезпечення цілісності даних використовувалися первинні та зовнішні ключі. Це дозволяє запобігти появі помилкових записів та забезпечує правильність взаємозв’язків між таблицями.

Наприклад, таблиця «Grades» пов’язана з таблицями «Students», «Teachers», «Subjects» та «Semesters».

Таблиця «TeacherAssignments» пов’язує між собою «Teachers», «Subjects» і «SchoolClasses».

Таблиця «ParentStudents» забезпечує зв’язок між «Parents» та «Students». Завдяки цьому вся структура бази даних відображає реальні процеси, що відбуваються в межах навчального закладу.

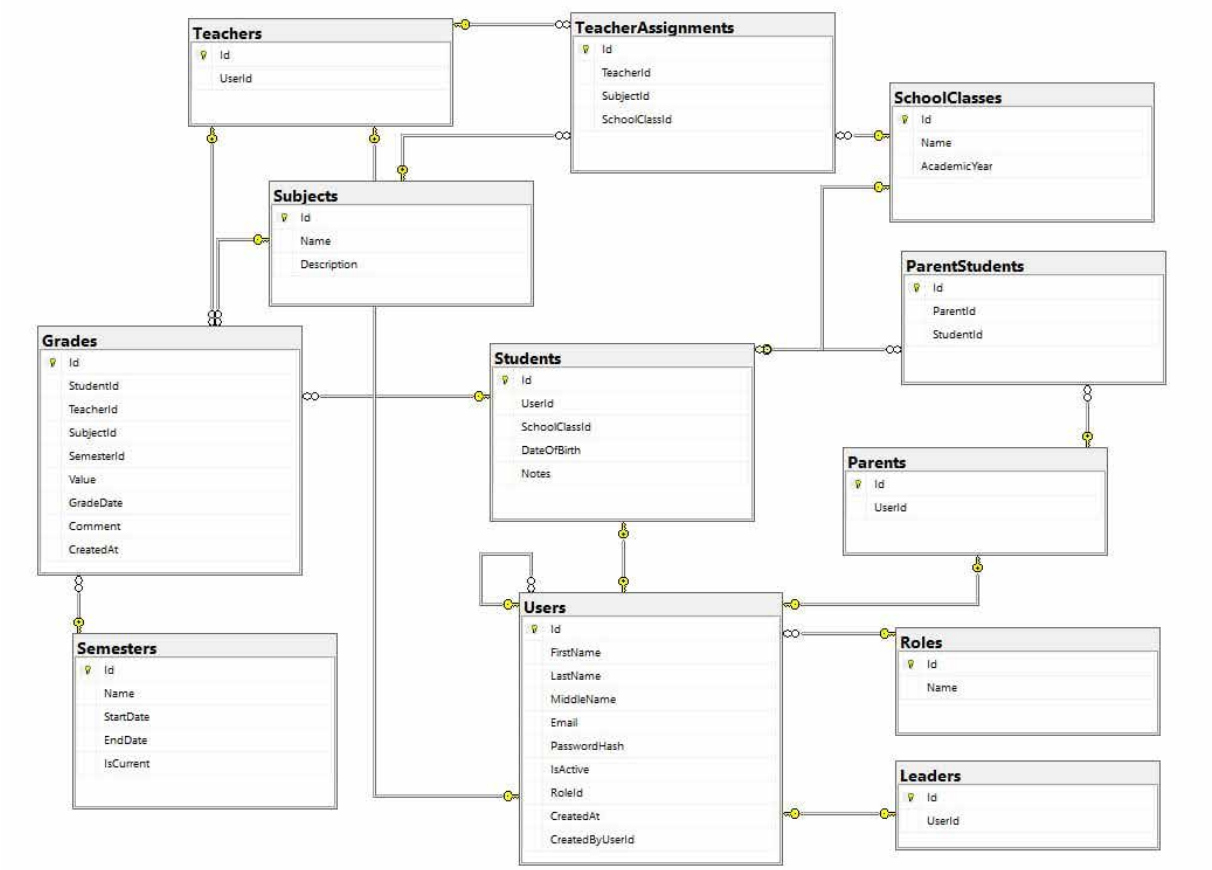


Рис. 2.3 Схема зв’язків між таблицями бази даних

Під час створення інформаційної бази також було реалізовано механізми перевірки коректності введення інформації. Для окремих полів встановлювалися типи даних, обмеження на допустимі значення, довжину тексту та логіку обов’язковості. Наприклад, для оцінок використовуються допустимі числові значення, для текстових полів визначено максимальну довжину, а для зв’язків між сутностями забезпечено перевірку коректності зовнішніх ключів.

Важливим етапом стало налаштування індексів для основних таблиць бази даних. Індекси дозволяють прискорити виконання SQL-запитів та

забезпечують швидкий пошук інформації навіть при значному обсязі даних. Особливо важливими є індекси для таблиць користувачів, учнів, оцінок, класів і таблиць, що використовуються для аналітичних запитів.

Після створення структури бази даних було виконано її наповнення тестовою інформацією. До системи було додано дані про користувачів, учнів, учителів, батьків, класи, навчальні предмети, семестри, шкільні події та результати навчання. Це дозволило перевірити правильність побудови взаємозв’язків між таблицями та протестувати роботу SQL-запитів і прикладної логіки системи.

Для взаємодії програмного забезпечення з інформаційною базою використовуються SQL-запити та засоби прикладного доступу до даних. За їх допомогою реалізується додавання нових записів, редагування інформації, пошук даних, фільтрація записів та формування статистики успішності.

Одним із важливих аспектів створення інформаційної бази є забезпечення захисту інформації. Оскільки система працює з персональними даними учнів і працівників навчального закладу, було передбачено механізми автентифікації, рольового доступу, контроль активності облікових записів та використання безпечного зберігання паролів у вигляді хешів.

Таким чином, у процесі створення інформаційної бази було реалізовано структуру таблиць, налаштовано взаємозв’язки між сутностями, забезпечено механізми цілісності та підготовлено основу для стабільної роботи інформаційної системи. Створена база даних забезпечує ефективне зберігання інформації та є фундаментом для реалізації функцій моніторингу успішності, аналітики та керування користувачами.

3 ПРИКЛАДНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Організаційна структура програмного забезпечення

Після завершення етапів аналізу предметної області та проєктування інформаційної бази наступним етапом розробки стало створення прикладного програмного забезпечення інформаційної системи моніторингу успішності учнів. Саме програмне забезпечення забезпечує взаємодію користувачів із базою даних, обробку інформації, формування статистики та реалізацію основних функціональних можливостей системи.

Організаційна структура програмного забезпечення визначає внутрішню будову системи, взаємодію окремих модулів та розподіл функцій між компонентами. Правильно побудована структура програмного забезпечення дозволяє забезпечити стабільну роботу системи, спростити процес розробки та створити можливість подальшого масштабування програмного продукту.

Інформаційна система моніторингу успішності учнів була реалізована за модульним принципом. Такий підхід дозволяє розділити програмне забезпечення на окремі функціональні компоненти, кожен із яких відповідає за виконання конкретних завдань. Використання модульної структури значно спрощує підтримку програмного забезпечення та дозволяє вносити зміни до окремих частин системи без впливу на інші компоненти.

Одним із головних компонентів системи є модуль авторизації користувачів. Даний модуль відповідає за перевірку облікових даних користувача та забезпечення безпечного доступу до функціональних можливостей системи. Після введення логіна та пароля система виконує перевірку даних та визначає рівень доступу відповідно до ролі користувача.

Інтерфейс авторизації користувача наведено на рисунку 3.1.

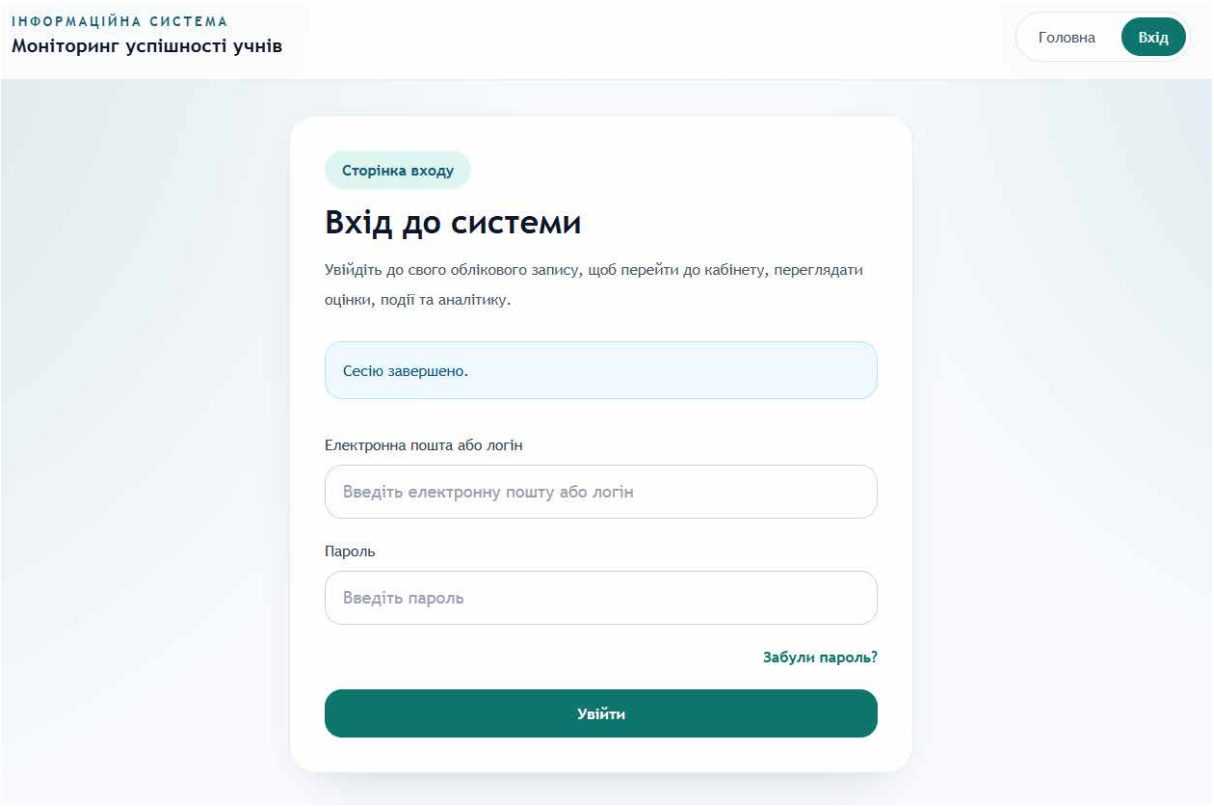


Рис. 3.1 Вікно авторизації користувача

Після успішної авторизації користувач отримує доступ до головного меню системи.

Одним із важливих компонентів програмного забезпечення є модуль адміністрування системи. Даний модуль призначений для керування основними даними інформаційної системи та забезпечує можливість створення нових користувачів, додавання класів і підтримки структури навчального закладу.

Адміністратор системи має можливість створювати нові облікові записи користувачів, визначати їх роль у системі та редагувати основну інформацію. Це дозволяє забезпечити контроль доступу до функціональних можливостей програмного забезпечення та організувати роботу користувачів відповідно до їх прав доступу.

Інтерфейс додавання нового користувача наведено на рисунку 3.2.

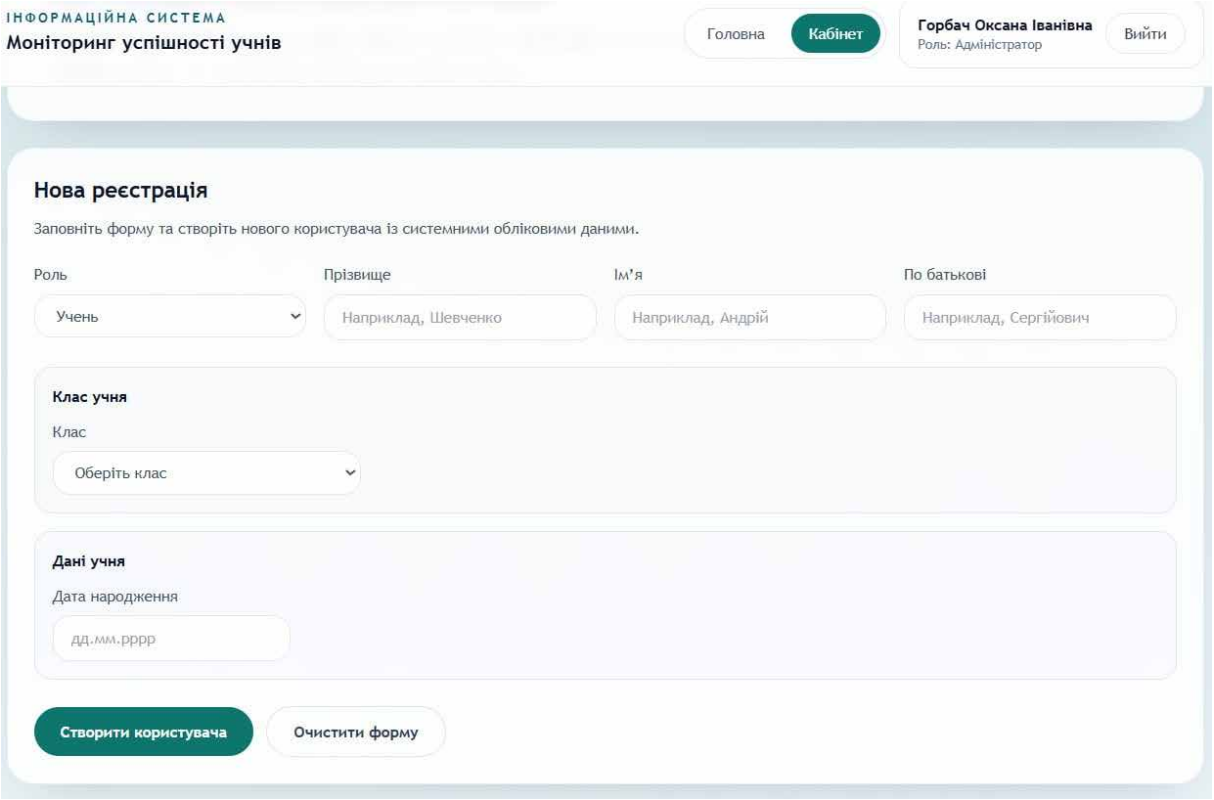


Рис. 3.2 Інтерфейс додавання нового користувача

Крім роботи з обліковими записами, адміністратор системи може створювати нові класи та редагувати інформацію про вже існуючі. Реалізація даного функціоналу дозволяє підтримувати актуальну структуру навчального закладу та забезпечує коректну організацію інформації у базі даних.

Інтерфейс додавання нового класу наведено на рисунку 3.3.

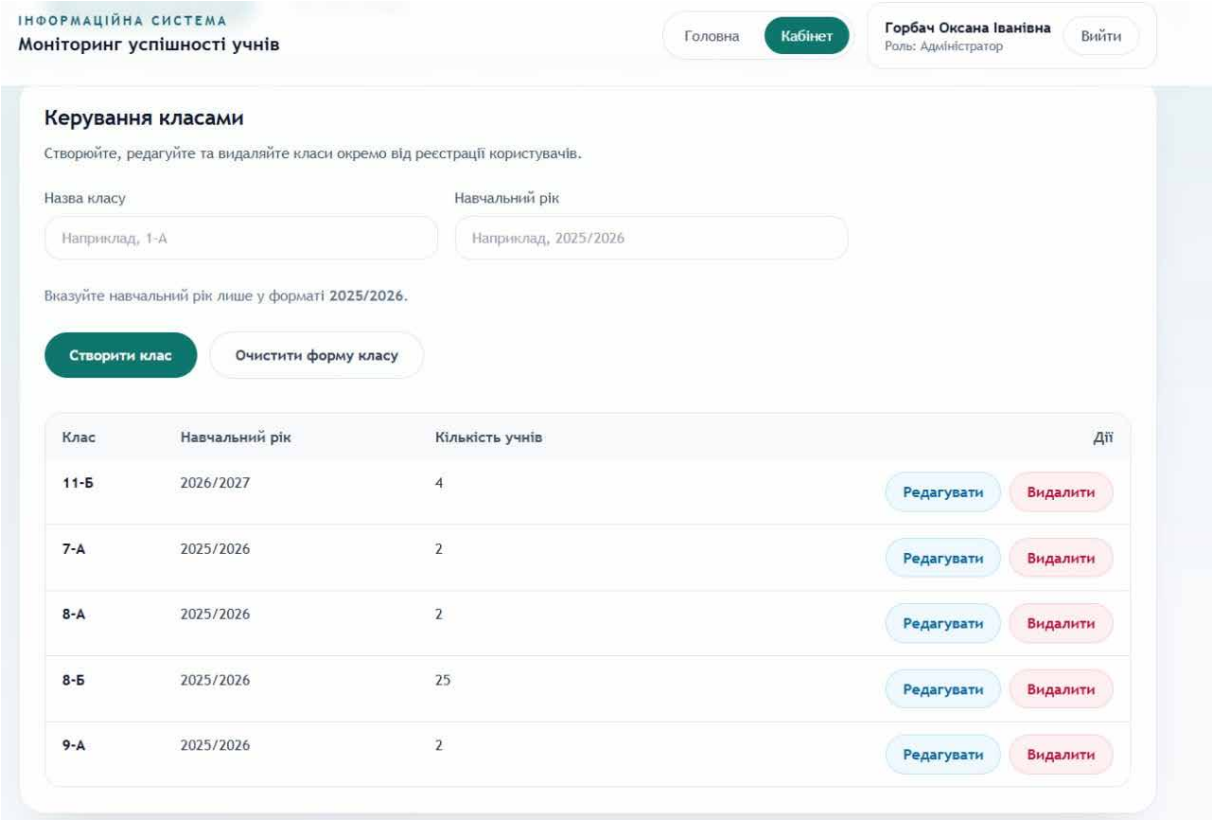


Рис. 3.3 Інтерфейс додавання нового класу

Для забезпечення зручності роботи у системі реалізовано механізми пошуку та фільтрації інформації, що дозволяє швидко знаходити необхідні записи та спрощує адміністрування інформаційної системи.

Наступним важливим компонентом є модуль електронного журналу. Саме цей модуль використовується для внесення та редагування оцінок учнів. Учитель має можливість обрати клас, предмет та конкретного учня, після чого система дозволяє додати нову оцінку або змінити вже існуючу.

Після збереження інформації система автоматично оновлює статистику успішності та виконує перерахунок середнього балу. Це дозволяє значно спростити процес ведення навчальної документації та зменшити кількість помилок під час обробки інформації.

Інтерфейс журналу оцінок наведено на рисунку 3.4.

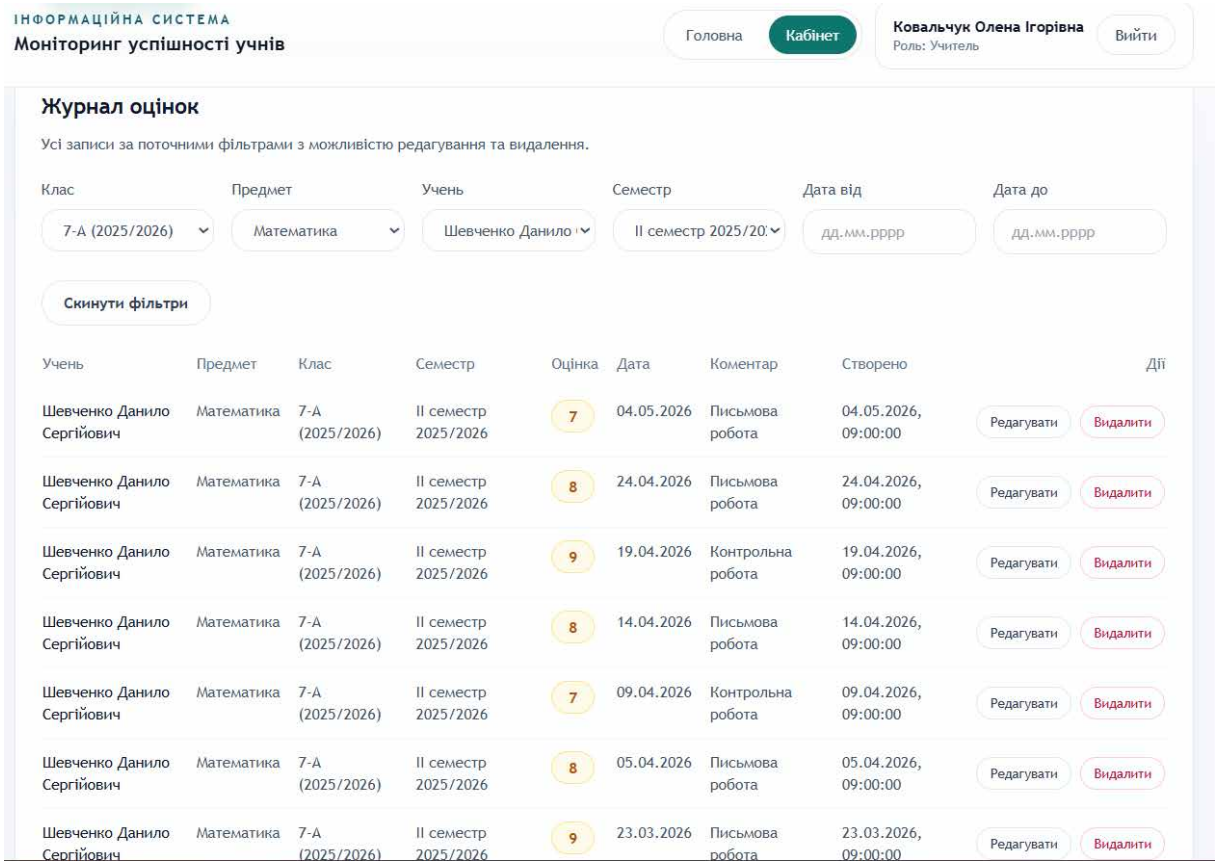


Рис. 3.4 Журнал оцінок

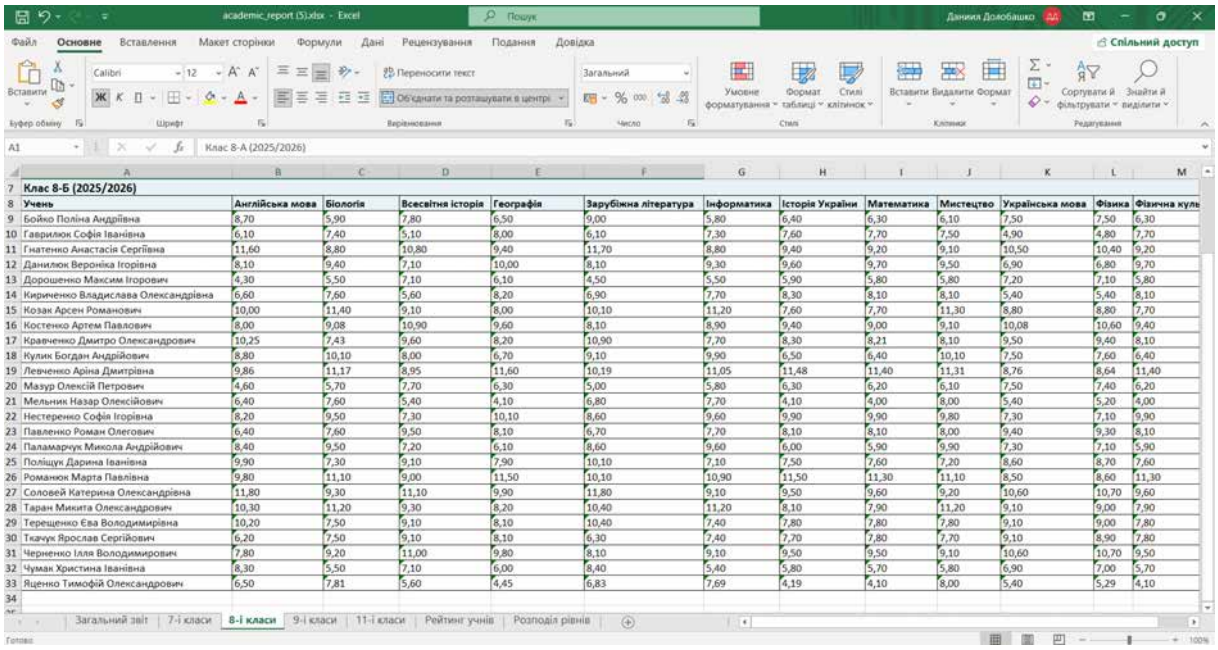
Для підвищення ефективності аналізу результатів навчання у системі реалізовано модуль статистики та аналітики. Даний компонент забезпечує автоматичне формування таблиць та діаграм, які дозволяють аналізувати успішність учнів протягом певного періоду часу.

За допомогою модуля аналітики користувач може переглядати середній бал окремого учня, результати класу або статистику з конкретного навчального предмета. Це дозволяє швидко виявляти проблеми у навчальному процесі та приймати відповідні рішення.

Окрему увагу було приділено модулю формування звітів. Система повинна забезпечувати автоматичне створення звітності щодо результатів навчання учнів, статистики класів та загальної успішності навчального закладу.

Користувач може сформувати звіт за певний період часу, вибрати клас або конкретного учня, після чого система автоматично виконує обробку інформації та формує готовий документ.

Приклад сформованого звіту наведено на рисунках 3.5 та 3.6.

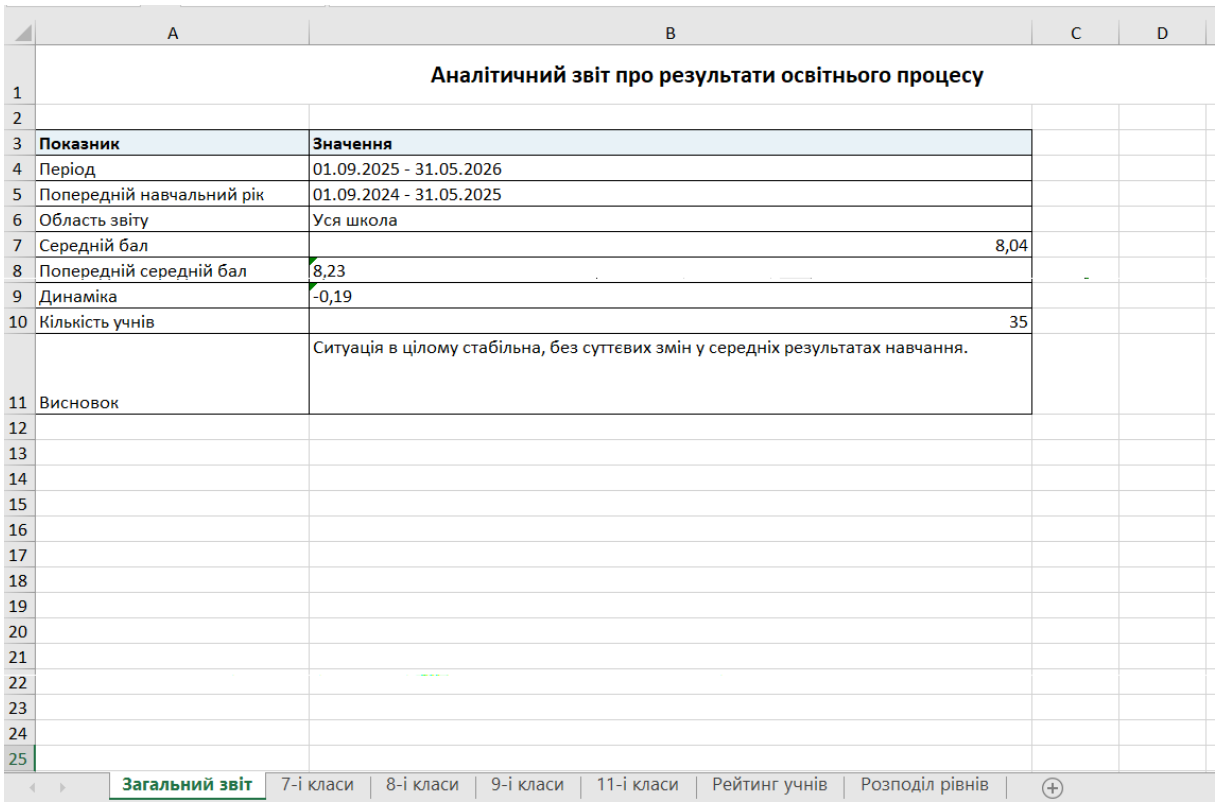


Клас 8-Б (2025/2026)

Учень	Англійська мова	Біологія	Всесвітня історія	Географія	Зарубіжна література	Інформатика	Історія України	Математика	Мистецтво	Українська мова	Фізика	Фізична куль
Бойко Поліна Андріївна	8,70	5,90	7,80	6,50	9,00	5,80	6,40	6,30	6,10	7,50	7,50	6,30
Гаврилюк Софія Іванівна	6,10	7,40	5,10	8,00	6,10	7,30	7,60	7,70	7,50	4,90	4,80	7,70
Гнатенко Анастасія Сергіївна	11,60	8,80	10,80	9,40	11,70	8,80	9,40	9,20	9,10	10,50	10,40	9,20
Данилюк Вероніка Ігорівна	8,10	9,40	7,10	10,00	8,10	9,30	9,60	9,70	9,50	6,90	6,80	9,70
Дорошенко Максим Ігорович	4,30	5,50	7,10	6,10	4,50	5,50	5,90	5,80	5,80	7,20	7,10	5,80
Кириченко Владислава Олександрівна	6,60	7,60	5,60	8,20	6,90	7,70	8,30	8,10	8,10	5,40	5,40	8,10
Козак Арсен Романович	10,00	11,40	9,10	8,00	10,10	11,20	7,60	7,70	11,30	8,80	8,80	7,70
Костенко Артем Павлович	8,00	9,08	10,90	9,60	8,10	8,90	9,40	9,00	9,10	10,08	10,60	9,40
Кравченко Дмитро Олександрович	10,25	7,43	9,60	8,20	10,90	7,70	8,30	8,21	8,10	9,50	9,40	8,10
Кулик Богдан Андрійович	8,80	10,10	8,00	6,70	9,10	9,90	6,50	6,40	10,10	7,50	7,60	6,40
Левченко Аріна Дмитрівна	9,86	11,17	8,95	11,60	10,19	11,05	11,48	11,40	11,31	8,76	8,64	11,40
Мазур Олексій Петрович	4,60	5,70	7,70	6,30	5,00	5,80	6,30	6,20	6,10	7,50	7,40	6,20
Мельник Назар Олексійович	6,40	7,60	5,40	4,10	6,80	7,70	4,10	4,00	8,00	5,40	5,20	4,00
Нестеренко Софія Ігорівна	8,20	9,50	7,30	10,10	8,60	9,60	9,90	9,90	9,80	7,30	7,10	9,90
Павленко Роман Олегович	6,40	7,60	9,50	8,10	6,70	7,70	8,10	8,10	8,00	9,40	9,30	8,10
Паламарчук Микола Андрійович	8,40	9,50	7,20	6,10	8,60	9,60	6,00	5,90	9,90	7,30	7,10	5,90
Поліщук Дарина Іванівна	9,90	7,30	9,10	7,90	10,10	7,10	7,50	7,60	7,20	8,60	8,70	7,60
Романюк Марта Павлівна	9,80	11,10	9,00	11,50	10,10	10,90	11,50	11,30	11,10	8,50	8,60	11,30
Соловей Катерина Олександрівна	11,80	9,30	11,10	9,90	11,80	9,10	9,50	9,60	9,20	10,60	10,70	9,60
Таран Микита Олександрович	10,30	11,20	9,30	8,20	10,40	11,20	8,10	7,90	11,20	9,10	9,00	7,90
Терещенко Єва Володимирівна	10,20	7,50	9,10	8,10	10,40	7,40	7,80	7,80	7,80	9,10	9,00	7,80
Ткачук Ярослав Сергійович	6,20	7,50	9,10	8,10	6,30	7,40	7,70	7,80	7,70	9,10	8,90	7,80
Черненко Ілля Володимирович	7,80	9,20	11,00	9,80	8,10	9,10	9,50	9,50	9,10	10,60	10,70	9,50
Чумак Христина Іванівна	8,30	5,50	7,10	6,00	8,40	5,40	5,80	5,70	5,80	6,90	7,00	5,70
Яценко Тимофій Олександрович	6,50	7,81	5,60	4,45	6,83	7,69	4,19	4,10	8,00	5,40	5,29	4,10

Загальний звіт | 7-і класи | 8-і класи | 9-і класи | 11-і класи | Рейтинг учнів | Розподіл рівнів

Рис. 3.5 Приклад звіту по окремому класу



Аналітичний звіт про результати освітнього процесу

Показник	Значення
Період	01.09.2025 - 31.05.2026
Попередній навчальний рік	01.09.2024 - 31.05.2025
Область звіту	Уся школа
Середній бал	8,04
Попередній середній бал	8,23
Динаміка	-0,19
Кількість учнів	35
Висновок	Ситуація в цілому стабільна, без суттєвих змін у середніх результатах навчання.

Загальний звіт | 7-і класи | 8-і класи | 9-і класи | 11-і класи | Рейтинг учнів | Розподіл рівнів

Рис. 3.6 Приклад загального звіту по школі

Під час розробки програмного забезпечення особливу увагу було приділено інтерфейсу користувача. Інтерфейс системи повинен бути простим, інтуїтивно зрозумілим та зручним у використанні. Оскільки системою користуватимуться люди з різним рівнем цифрової підготовки, важливо забезпечити максимально просту навігацію та логічне розташування елементів керування.

Для покращення зручності роботи в інтерфейсі використовуються таблиці, кнопки швидкого доступу, меню навігації та форми введення інформації. Такий підхід дозволяє спростити взаємодію користувача із системою та підвищити ефективність роботи з програмним забезпеченням.

Таким чином, організаційна структура програмного забезпечення дозволяє забезпечити стабільну роботу системи, ефективну взаємодію між модулями та зручну роботу користувачів із програмним забезпеченням інформаційної системи моніторингу успішності учнів.

Після визначення загальної архітектури системи було виконано проєктування внутрішньої структури програмного забезпечення. Для цього програмне забезпечення було розділено на окремі компоненти, кожен із яких відповідає за виконання певного набору функцій.

Клієнтська частина системи забезпечує взаємодію користувача з програмним забезпеченням через графічний інтерфейс та механізми передачі запитів до серверної частини. Серверна частина відповідає за обробку даних, виконання бізнес-логіки та взаємодію з базою даних.

Для наочного представлення структури програмного забезпечення та взаємозв’язків між його компонентами була побудована діаграма компонентів.

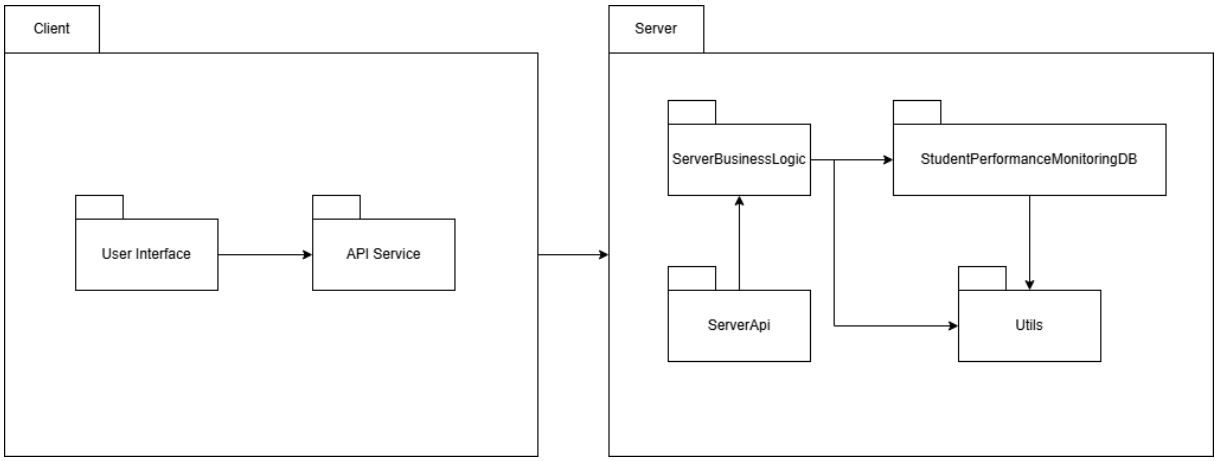


Рис. 3.7 Діаграма компонентів

Як показано на діаграмі, клієнтська частина містить інтерфейс користувача та модуль API Service, через який здійснюється взаємодія із сервером. Серверна частина системи містить модулі бізнес-логіки, серверний API та базу даних StudentPerformanceMonitoringDB. Така структура дозволяє розділити функції між компонентами системи та забезпечити стабільну взаємодію між ними.

3.2 Вибір інструментарію для створення ППЗ

Під час розробки інформаційної системи моніторингу успішності учнів важливим етапом є вибір інструментів та технологій, які будуть використовуватися для створення програмного забезпечення. Саме правильно обраний інструментарій дозволяє забезпечити стабільну роботу системи, зручність розробки, можливість масштабування та підтримку сучасного інтерфейсу користувача.

Під час вибору технологій враховувалися функціональні вимоги до системи, необхідність роботи з базою даних, підтримка графічного інтерфейсу та можливість швидкої обробки інформації. Також важливими критеріями були стабільність роботи, сумісність із сучасними операційними системами та підтримка подальшого розвитку програмного забезпечення.

Для створення інформаційної системи було використано мову програмування C# для серверної частини та TypeScript для клієнтської частини. Такий підхід дозволив поєднати надійні засоби екосистеми .NET із сучасними вебтехнологіями побудови користувацького інтерфейсу.

Серверну частину реалізовано на платформі .NET 8 із використанням ASP.NET Core Web API. Дана технологія забезпечує створення веборієнтованого серверного застосунку, підтримку REST-взаємодії, реалізацію бізнес-логіки та інтеграцію з базою даних Microsoft SQL Server.

Для розробки клієнтської частини застосовано бібліотеку React, мову TypeScript та інструмент збірки Vite. Використання компонентного підходу дало можливість побудувати зручний інтерфейс користувача, а TypeScript забезпечив кращу структурованість і надійність клієнтського коду.

Важливою перевагою обраного інструментарію є зручність розробки та налагодження вебінтерфейсу. Компонентна архітектура React дозволяє швидко створювати сторінки, таблиці, форми введення, графіки та інші елементи інтерфейсу без дублювання коду.

Для зберігання інформації у системі було використано Microsoft SQL Server. Основна взаємодія серверної частини з базою даних реалізована за допомогою Entity Framework Core, що забезпечує зручну роботу з сутностями, LINQ-запитами та міграціями бази даних.

Окремі технічні операції масового запису даних у системі виконуються через ADO.NET та Microsoft.Data.SqlClient, однак це не є основним механізмом доступу до даних. Базові операції отримання, додавання, редагування та видалення інформації реалізовано переважно через Entity Framework Core.

Для виконання запитів до бази даних використовувалася мова SQL. Вона дозволяє створювати таблиці, змінювати структуру бази даних, додавати записи та виконувати пошук інформації.

Під час вибору інструментарію також враховувалися питання продуктивності та стабільності роботи системи. Використання платформи .NET та SQL Server дозволяє забезпечити швидке виконання основних операцій навіть при великій кількості записів у базі даних.

Особливу увагу було приділено реалізації інтерфейсу користувача. Оскільки системою користуватимуться учителі, учні, батьки та адміністрація навчального закладу, інтерфейс повинен бути максимально простим та зрозумілим.

Під час розробки програмного забезпечення також використовувалися механізми обробки помилок. Це дозволяє запобігти аварійному завершенню роботи програми у випадку неправильного введення даних або помилок підключення до бази даних.

Для підвищення безпеки системи було реалізовано механізми авторизації користувачів та розмежування прав доступу. Кожен користувач отримує доступ лише до тих функцій, які відповідають його ролі у системі.

Окрему увагу було приділено можливості подальшого розширення функціоналу програмного забезпечення. Обрані технології дозволяють у майбутньому додати нові модулі без необхідності повного перепроєктування системи.

Таким чином, у результаті аналізу сучасних технологій для розробки інформаційної системи моніторингу успішності учнів було обрано стек C#, .NET 8, ASP.NET Core Web API, React, TypeScript, Vite та Microsoft SQL Server. Використання даного інструментарію дозволяє забезпечити стабільну роботу вебзастосунку, зручний інтерфейс користувача та ефективну взаємодію з інформаційною базою.

3.3 Алгоритмізація та програмування програмних модулів

Під час розробки інформаційної системи моніторингу успішності учнів особлива увага приділялася алгоритмізації основних функціональних процесів. Побудова алгоритмів дозволяє визначити послідовність виконання операцій, забезпечити правильну взаємодію між модулями системи та реалізувати стабільну роботу програмного забезпечення.

На основі функціональних вимог системи було розроблено алгоритми роботи основних програмних модулів. Найбільш важливими серед них є алгоритм авторизації користувача, алгоритм роботи електронного журналу та алгоритм формування статистики успішності.

Одним із основних модулів інформаційної системи є модуль авторизації користувачів. Його головним призначенням є перевірка облікових даних користувача та надання доступу до функціональних можливостей системи відповідно до ролі користувача.

Робота алгоритму починається із відкриття сторінки авторизації. Користувач вводить логін та пароль у відповідні поля та натискає кнопку входу.

Після цього система перевіряє, чи всі поля були заповнені. Якщо хоча б одне поле є порожнім, система відображає повідомлення про необхідність введення даних та повертає користувача до форми авторизації.

У випадку правильного заповнення полів система виконує перевірку введених облікових даних через серверну частину. Якщо користувача знайдено у базі даних, система визначає його роль та відкриває відповідний кабінет або сторінку з потрібним рівнем доступу.

Якщо введені дані є неправильними, система відображає повідомлення про помилку та блокує доступ до програмного забезпечення.

Одним із ключових елементів системи є модуль електронного журналу, який забезпечує роботу з оцінками учнів. Основним завданням

даного модуля є внесення, редагування та збереження результатів навчальної діяльності.

Робота алгоритму починається з вибору користувачем необхідного класу та навчального предмета. Після цього система виконує SQL-запит до бази даних та завантажує список учнів відповідного класу.

Далі користувач обирає необхідного учня та вводить оцінку у форму електронного журналу. Перед збереженням інформації система перевіряє правильність введених даних. Якщо оцінка введена неправильно або поле залишилося порожнім, система відображає повідомлення про помилку та не дозволяє виконати збереження.

Після успішної перевірки даних система виконує SQL-запит для додавання або оновлення інформації у таблиці оцінок. Після збереження даних система автоматично оновлює інформацію в електронному журналі та виконує перерахунок середнього балу учня.

Завершальним етапом алгоритму є оновлення відображення інформації у таблиці оцінок та повернення користувача до форми електронного журналу.

Важливим компонентом інформаційної системи є модуль статистики успішності, який забезпечує автоматичний аналіз результатів навчання учнів та відображення статистичних показників у графічному вигляді.

Робота алгоритму починається з вибору користувачем необхідного класу, предмета або учня для аналізу інформації. Після цього система виконує SQL-запит до бази даних та отримує необхідні дані про оцінки.

Далі система виконує обробку отриманої інформації та обчислює середній бал, кількість оцінок і загальні статистичні показники. Після завершення обчислень результати передаються до модуля візуалізації.

На наступному етапі система формує таблиці, графіки або діаграми для відображення результатів успішності. Отримана інформація відображається на відповідній аналітичній сторінці.

Якщо у базі даних відсутня необхідна інформація, система відображає повідомлення про відсутність даних для аналізу.

Після завершення формування статистики користувач отримує можливість перегляду результатів або повернення до головного меню системи.

Алгоритм формування статистики успішності наведено на рисунку 3.8.

Рис. 3.8 Алгоритм формування статистики успішності учня

Реалізація наведених алгоритмів дозволила забезпечити стабільну роботу інформаційної системи, правильну взаємодію користувачів із базою даних та автоматизацію основних процесів контролю успішності учнів. Побудовані алгоритми стали основою програмної реалізації функціональних модулів та забезпечили ефективну роботу системи в умовах експлуатації.

Під час роботи системи важливе значення мають алгоритми завантаження та збереження даних. Алгоритм завантаження забезпечує перевірку наявності необхідних компонентів, зчитування інформації з бази даних, передачу отриманих даних до клієнтської частини системи та оновлення інтерфейсу користувача.

Рисунок 3.9 – Блок-схема алгоритму завантаження даних із бази даних

Після завантаження інформації користувач може виконувати необхідні дії в системі. У випадку додавання або редагування записів використовується алгоритм збереження даних, який передбачає отримання інформації з інтерфейсу, формування структурованого набору даних, передачу його на серверну частину системи та запис у базу даних.

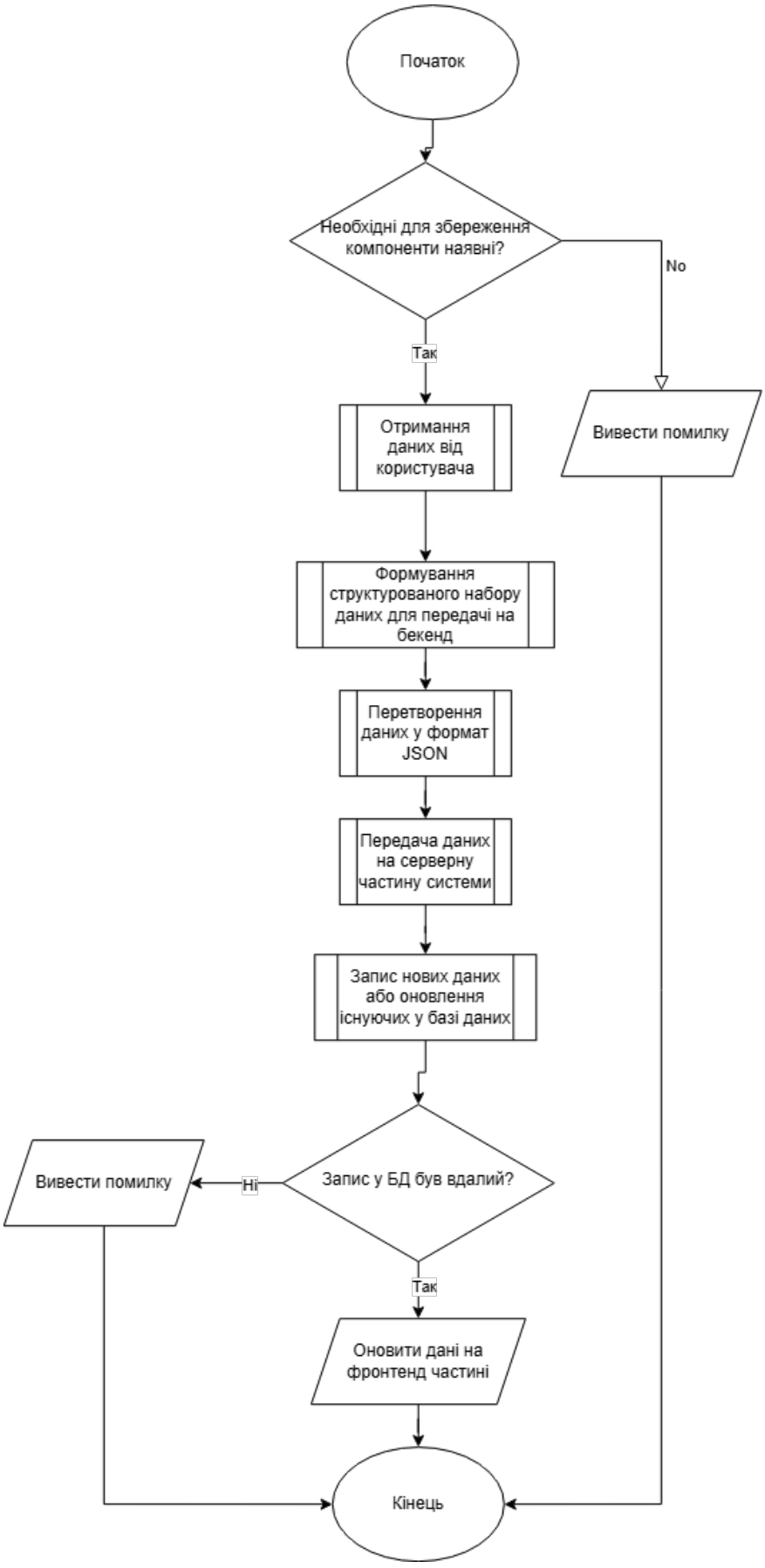


Рисунок 3.10 – Блок-схема алгоритму збереження даних у базі даних

Під час проєктування програмного забезпечення важливим етапом стало визначення структури взаємодії між основними компонентами інформаційної системи. Оскільки система працює за клієнт-серверним принципом, необхідно було організувати взаємодію між користувацьким інтерфейсом, серверною частиною застосунку та базою даних.

Користувач працює із системою через браузер та frontend-застосунок. Передача даних між клієнтською та серверною частинами здійснюється через HTTP/HTTPS із використанням формату JSON. Сервер застосунку містить основні програмні модулі системи, серед яких модуль авторизації, API, модуль аналітики та модуль роботи з оцінками. Уся інформація зберігається на сервері бази даних Microsoft SQL Server 2022 у базі StudentPerformanceMonitoringDb.

Таким чином, використання клієнт-серверної архітектури дозволяє забезпечити централізоване зберігання інформації, стабільну взаємодію між компонентами системи та можливість подальшого масштабування програмного забезпечення.

Під час розробки інформаційної системи моніторингу успішності учнів важливу увагу було приділено забезпеченню зручності користування не лише на персональних комп’ютерах, а й на мобільних пристроях. Оскільки сучасні користувачі часто взаємодіють із вебресурсами за допомогою смартфонів і планшетів, система повинна коректно відображатися на екранах різного розміру та забезпечувати збереження функціональності незалежно від типу пристрою. Саме тому під час реалізації клієнтської частини було враховано принципи адаптивного вебдизайну.

Адаптивність інтерфейсу полягає в автоматичному пристосуванні структури сторінки, розмірів елементів, відступів, шрифтів та розташування функціональних блоків до поточної ширини екрана. Це дозволяє уникнути ситуацій, коли користувачу доводиться постійно масштабувати сторінку,

прокручувати її в різних напрямках або втрачати доступ до частини функцій через некоректне відображення елементів. У створеній системі адаптація реалізована таким чином, щоб на мобільних пристроях основні елементи інтерфейсу зберігали читабельність, логічну структуру та зручність взаємодії.

Особливу роль адаптивність відіграє для сторінок авторизації, кабінетів користувачів, сторінок аналітики та електронного журналу. Наприклад, на сторінці входу поля введення, кнопки та інформаційні повідомлення мають залишатися достатньо великими для комфортного використання на сенсорному екрані. На сторінках із таблицями та аналітичними блоками важливо забезпечити таке розташування компонентів, за якого користувач зможе швидко переглядати ключову інформацію навіть за обмеженої ширини екрана. Для цього окремі блоки можуть розташовуватися один під одним, а ширина контейнерів і відступи автоматично змінюються відповідно до поточного розміру вікна браузера.

Реалізація адаптивного інтерфейсу позитивно впливає не лише на зовнішній вигляд системи, а й на загальну якість взаємодії користувача із застосунком. Учні, батьки, вчителі та керівництво закладу освіти можуть переглядати інформацію, результати навчання, аналітичні показники та інші дані не лише з настільного комп’ютера, а й із мобільного телефону. Це підвищує доступність системи, робить її більш універсальною та відповідає сучасним вимогам до веборієнтованих інформаційних рішень.

Таким чином, адаптивність є важливою характеристикою розробленої системи, оскільки забезпечує її коректне функціонування та зручність використання на різних типах пристроїв. Це особливо важливо для освітнього середовища, де користувачі можуть звертатися до системи в різних умовах і з різних платформ.

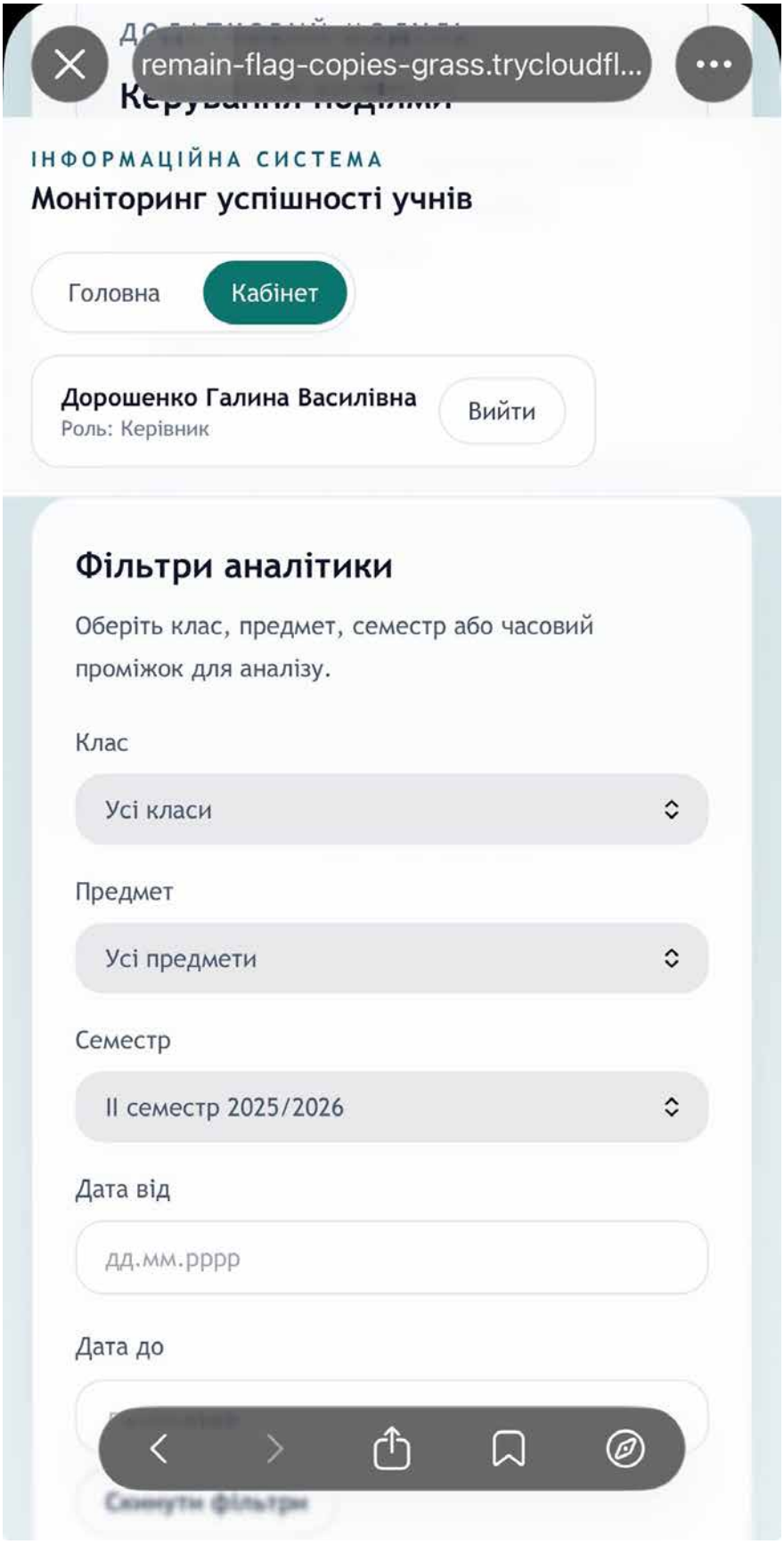


Рис. 3.11 Адаптивне відображення інтерфейсу інформаційної системи на екрані мобільного пристрою

4 РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ СИСТЕМИ

4.1 Тестування системи

Після завершення розробки інформаційної системи моніторингу успішності учнів було проведено комплексне тестування програмного забезпечення. Основною метою тестування є перевірка правильності роботи функціональних модулів, стабільності роботи системи, коректності взаємодії з базою даних та відповідності програмного забезпечення поставленим вимогам.

Тестування є важливим етапом життєвого циклу програмного забезпечення, оскільки дозволяє виявити помилки ще до впровадження системи у реальні умови експлуатації. Під час тестування оцінювалася правильність виконання операцій, швидкість роботи програмного забезпечення, зручність інтерфейсу користувача та стійкість системи до помилок.

У процесі тестування перевірялися такі основні компоненти системи:

– модуль авторизації користувачів;

– модуль роботи з учнями;

– електронний журнал;

– модуль статистики успішності;

– модуль оголошень та подій;

– модуль формування звітів;

– взаємодія з базою даних;

– механізми обробки помилок.

На першому етапі було проведено функціональне тестування системи. Даний вид тестування дозволяє перевірити правильність роботи окремих

функцій програмного забезпечення та відповідність отриманих результатів очікуваним значенням.

Одним із перших тестувався модуль авторизації користувачів. Під час перевірки тестувалася правильність введення логіна та пароля, робота механізму розмежування прав доступу та реакція системи на неправильне введення даних.

Було перевірено:

– вхід користувача з правильними обліковими даними;

– блокування доступу при неправильному паролі;

– роботу різних ролей користувачів;

– відкриття відповідних модулів після авторизації.

У результаті тестування було встановлено, що система коректно виконує перевірку користувачів та забезпечує доступ лише до дозволених функцій.

Результат тестування модуля авторизації наведено на рисунках 4.1 та 4.2.

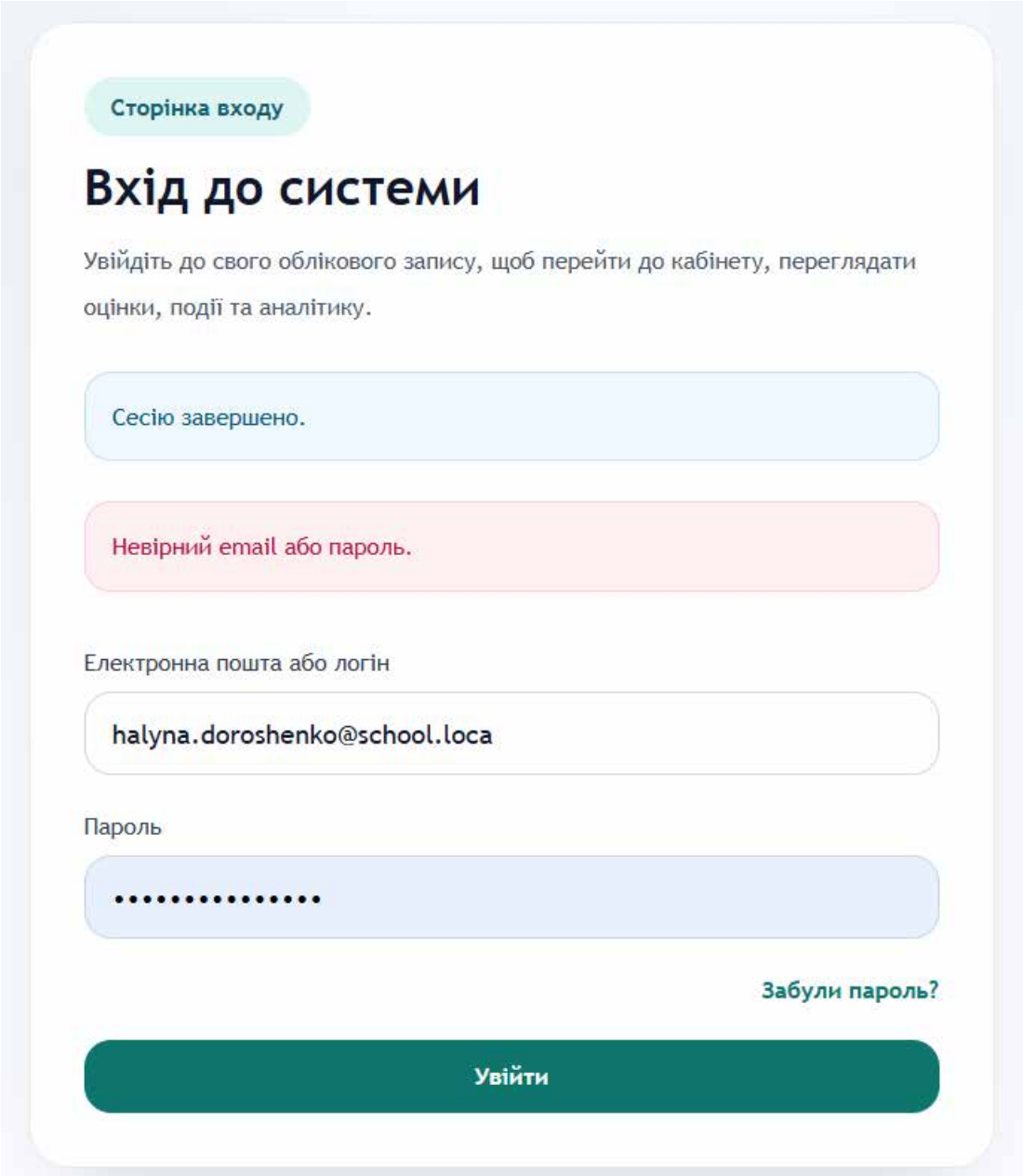


Рис. 4.1 Перевірка логіну та паролю на наявність в базі даних

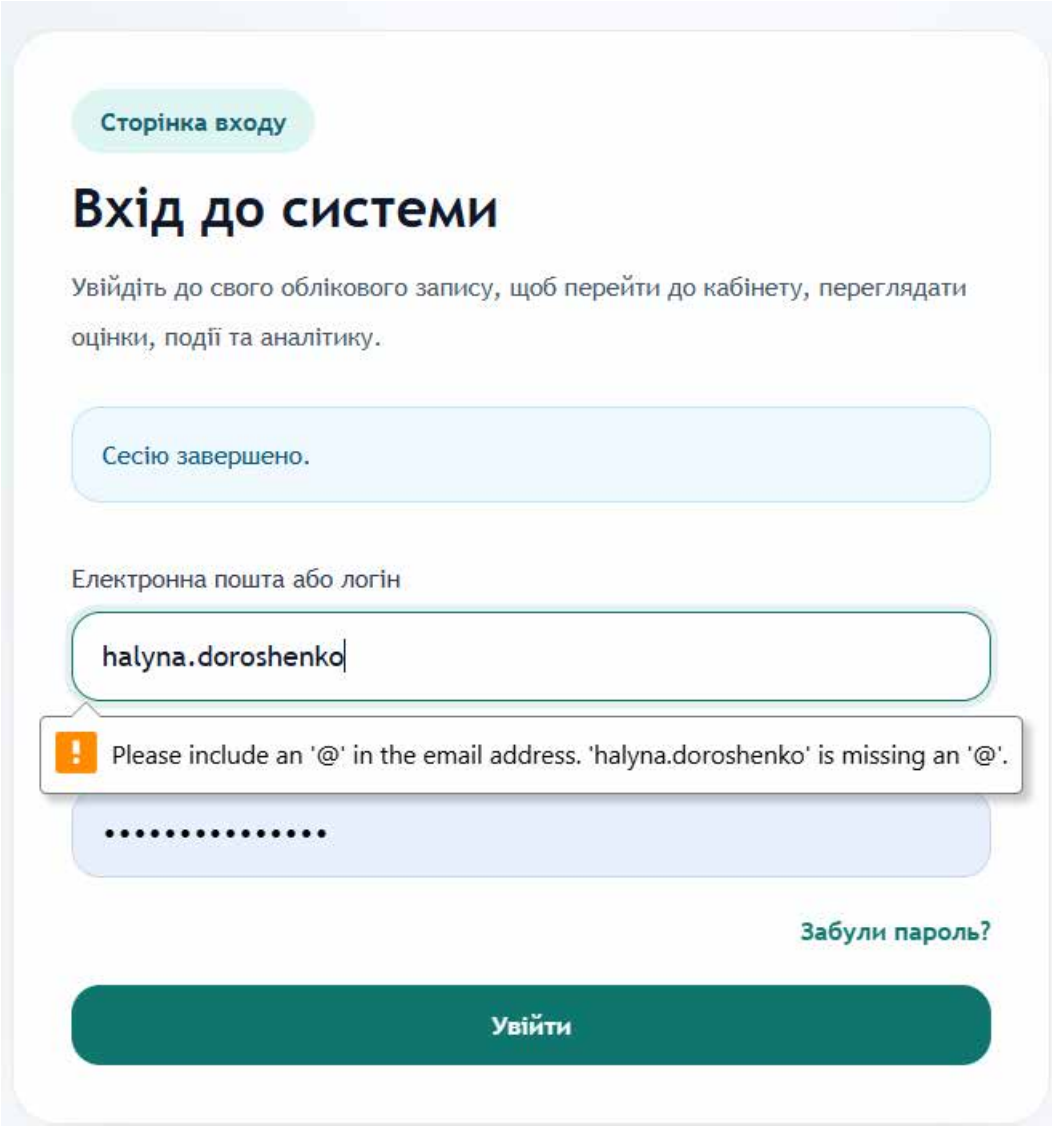


Рис. 4.2 Перевірка логіну на наявність ключових символів пошти

Наступним етапом стало тестування електронного журналу. Даний модуль є одним із ключових компонентів системи, оскільки забезпечує внесення та перегляд оцінок учнів.

Під час тестування перевірялася можливість:

– додавання оцінок;

– редагування інформації;

– збереження даних у базі даних;

– перегляду результатів навчання;

– автоматичного оновлення статистики.

У процесі тестування було встановлено, що система коректно виконує операції з оцінками та автоматично оновлює інформацію після внесення змін.

Окрему увагу було приділено тестуванню взаємодії програмного забезпечення з базою даних Microsoft SQL Server. Для цього перевірялася правильність виконання SQL-запитів, швидкість пошуку інформації та стабільність роботи системи при обробці великої кількості записів.

Під час тестування виконувалися:

– додавання нових записів;

– редагування інформації;

– видалення записів;

– пошук учнів;

– формування статистики;

– отримання інформації з бази даних.

Було встановлено, що система стабільно працює з інформаційною базою та забезпечує швидке виконання основних операцій.

Важливим етапом стало тестування модуля статистики успішності. Даний компонент відповідає за автоматичний аналіз результатів навчання та відображення статистичних показників у вигляді графіків і таблиць.

Під час тестування перевірялися:

– правильність обчислення середнього балу;

– формування статистики за предметами;

– відображення результатів навчання;

– побудова графіків успішності.

У результаті тестування було встановлено, що система коректно виконує всі обчислення та забезпечує правильне відображення статистичних даних.

Наступним етапом стало тестування модуля оголошень та подій. Даний компонент дозволяє керівнику навчального закладу створювати, редагувати та публікувати події для інших користувачів системи.

Під час тестування перевірялася:

– можливість створення оголошень;

– редагування інформації;

– відображення подій у системі;

– збереження даних у базі даних.

У результаті тестування було встановлено, що модуль коректно виконує всі основні функції та забезпечує відображення актуальної інформації для користувачів системи.

Окрему увагу під час тестування було приділено перевірці інтерфейсу користувача. Основною метою даного етапу є оцінка зручності використання програмного забезпечення та правильності розташування елементів керування.

Під час тестування перевірялися:

– логічність структури меню;

– швидкість переходу між формами;

– коректність відображення інформації;

– зручність введення даних;

– стабільність роботи елементів інтерфейсу.

У результаті тестування було встановлено, що інтерфейс системи є простим та зрозумілим для користувачів. Основні функції доступні через головне меню та окремі сторінки кабінетів, а елементи інтерфейсу мають логічну структуру.

Для забезпечення стабільної роботи системи також було проведено тестування обробки помилок. Під час перевірки система тестувалася в умовах неправильного введення інформації та нестандартних ситуацій.

Перевірялася реакція системи на:

– введення порожніх значень;

– введення неправильних оцінок;

– втрату з’єднання з базою даних;

– помилки підключення до SQL Server.

У результаті тестування було встановлено, що система коректно обробляє помилки та відображає відповідні повідомлення користувачеві без аварійного завершення роботи програми.

Важливим етапом стало тестування безпеки інформаційної системи. Оскільки система працює з персональними даними учнів та працівників навчального закладу, необхідно забезпечити захист інформації від несанкціонованого доступу.

Було перевірено правильність роботи механізмів:

– авторизації користувачів;

– розмежування прав доступу;

– захисту інформаційної бази;

– обробки помилок під час роботи із серверною частиною та базою даних.

Результати тестування підтвердили, що користувачі мають доступ лише до тих функцій, які відповідають їх ролі у системі.

Під час тестування також оцінювалася продуктивність програмного забезпечення. Перевірялася швидкість запуску системи, час відкриття сторінок та швидкість виконання SQL-запитів.

Крім цього, було проведено тестування формування аналітичних звітів та експорту результатів у формат Excel. Перевірялася коректність обчислення показників, формування структури звіту та створення файлу для подальшого використання. Результати тестування підтвердили правильність роботи механізмів формування звітів та експорту аналітичних даних.

Таким чином, у процесі тестування було перевірено роботу всіх основних компонентів інформаційної системи моніторингу успішності учнів. Результати тестування підтвердили правильність роботи програмного забезпечення, стабільність взаємодії з базою даних та відповідність системи поставленим функціональним вимогам.

4.2 Вимоги до апаратного та програмного забезпечення

Для стабільної та коректної роботи інформаційної системи моніторингу успішності учнів необхідно забезпечити відповідні апаратні та програмні умови. Правильно підібране технічне забезпечення дозволяє забезпечити високу швидкість роботи системи, стабільну взаємодію з базою даних та комфортну роботу користувачів.

Оскільки розроблена система є веборієнтованим клієнт-серверним програмним забезпеченням із підключенням до бази даних Microsoft SQL Server, основні вимоги стосуються продуктивності клієнтського комп’ютера, наявності сучасного браузера, доступності серверної частини та підтримки необхідного програмного середовища на сервері.

Під час визначення технічних вимог враховувалися особливості роботи системи, обсяг інформації, який обробляється програмним забезпеченням, а також можливість одночасної роботи кількох користувачів із базою даних.

Для коректної роботи клієнтської частини рекомендовано використовувати комп’ютер або ноутбук із сучасним процесором, який забезпечує комфортну роботу браузера та швидке опрацювання інтерфейсу. Мінімально допустимим варіантом є двоядерний процесор із тактовою частотою не менше 2 ГГц. Для більш комфортної роботи доцільно використовувати процесори рівня Intel Core i3, Intel Core i5 або їх аналоги.

Обсяг оперативної пам’яті також відіграє важливу роль у роботі клієнтської та серверної частин системи. Для стабільної роботи користувацького робочого місця рекомендований обсяг оперативної пам’яті становить не менше 8 ГБ. Для сервера застосунку та бази даних доцільно використовувати 8 ГБ оперативної пам’яті або більше.

Для зберігання серверної частини застосунку, зібраної клієнтської частини та бази даних необхідно забезпечити достатній обсяг вільного місця на накопичувачі. Мінімально рекомендований обсяг вільного місця становить 10 ГБ. Використання SSD-накопичувача дозволяє значно підвищити швидкість запуску системи та виконання операцій із базою даних.

Однією з основних вимог до програмного забезпечення є наявність сучасного веббраузера на робочому місці користувача. Найбільш стабільна робота клієнтської частини забезпечується в актуальних версіях Google Chrome, Microsoft Edge або інших браузерів із підтримкою сучасних вебстандартів.

Для роботи серверної частини необхідна наявність середовища виконання .NET 8 або опублікованої серверної збірки ASP.NET Core. Саме ця платформа забезпечує виконання програмного коду серверного застосунку та коректну взаємодію з базою даних.

Для роботи з інформаційною базою використовується система управління базами даних Microsoft SQL Server. Вона забезпечує зберігання інформації, виконання SQL-запитів та підтримку механізмів захисту даних.

Для забезпечення взаємодії між програмним забезпеченням та базою даних необхідно налаштувати підключення до SQL Server. У параметрах підключення вказуються адреса сервера, назва бази даних, логін та пароль користувача.

Особливу увагу під час визначення вимог було приділено питанням мережевої взаємодії. Якщо система використовується у межах локальної

мережі навчального закладу, сервер бази даних повинен бути доступний для всіх робочих станцій користувачів.

У випадку використання системи на одному комп’ютері база даних може бути розміщена локально. Якщо ж система використовується декількома користувачами одночасно, доцільно використовувати окремий сервер бази даних.

Для забезпечення стабільної роботи системи також необхідно враховувати питання інформаційної безпеки. Комп’ютери користувачів повинні бути захищені антивірусним програмним забезпеченням, а доступ до бази даних має здійснюватися лише авторизованими користувачами.

Крім цього, рекомендується періодично створювати резервні копії інформаційної бази штатними засобами Microsoft SQL Server. У поточній версії системи автоматизований модуль резервного копіювання не реалізовано, тому такі дії виконуються на рівні адміністрування СУБД.

Під час експлуатації системи також важливо контролювати стабільність роботи мережевого з’єднання та наявність доступу до сервера бази даних. У випадку втрати з’єднання система повинна коректно обробляти помилки та повідомляти користувача про проблеми з підключенням.

Важливим аспектом є зручність роботи користувачів із програмним забезпеченням. Робоче місце користувача повинно бути обладнане монітором із достатньою роздільною здатністю для комфортної роботи з таблицями та формами введення інформації. Рекомендована роздільна здатність екрана становить не менше 1366×768 пікселів.

Для ефективної роботи системи також необхідно забезпечити стабільне електроживлення та захист обладнання від перепадів напруги. Використання джерела безперебійного живлення дозволяє запобігти втраті інформації у випадку аварійного вимкнення електроенергії.

Під час визначення вимог до програмного забезпечення також враховувалася можливість подальшого розвитку системи. Обрана архітектура дозволяє у майбутньому розширити функціонал програмного забезпечення та інтегрувати нові модулі без значних змін у вже реалізованій структурі системи.

Таким чином, визначені вимоги до апаратного та програмного забезпечення забезпечують стабільну роботу інформаційної системи моніторингу успішності учнів, швидку обробку інформації та комфортну роботу користувачів із програмним забезпеченням.

4.3 Склад інсталяційного пакету

Для забезпечення коректної роботи інформаційної системи моніторингу успішності учнів необхідно правильно організувати процес встановлення програмного забезпечення. Саме тому важливим етапом розробки є формування інсталяційного пакету, який містить усі необхідні компоненти для запуску та стабільної роботи системи.

Пакет розгортання являє собою набір серверних файлів, зібраної клієнтської частини, конфігураційних даних та допоміжних компонентів, необхідних для запуску інформаційної системи у клієнт-серверному середовищі. Правильно сформований пакет розгортання дозволяє спростити процес впровадження програмного забезпечення та мінімізувати ризик виникнення помилок під час налаштування системи.

Під час формування пакету розгортання особлива увага приділялася простоті налаштування. Адміністратор повинен мати можливість підготувати систему до роботи без складного ручного перепроєктування структури застосунку або зміни програмного коду.

До складу пакету розгортання входить опублікована серверна частина ASP.NET Core Web API, яка забезпечує запуск основної бізнес-логіки системи та взаємодію з базою даних Microsoft SQL Server.

Крім серверної частини, пакет містить зібрану клієнтську частину вебзастосунку, конфігураційні файли та допоміжні компоненти, необхідні для коректної роботи користувацького інтерфейсу та мережевої взаємодії між frontend і backend. Структуру пакету розгортання наведено на рисунках 4.3 та 4.4.

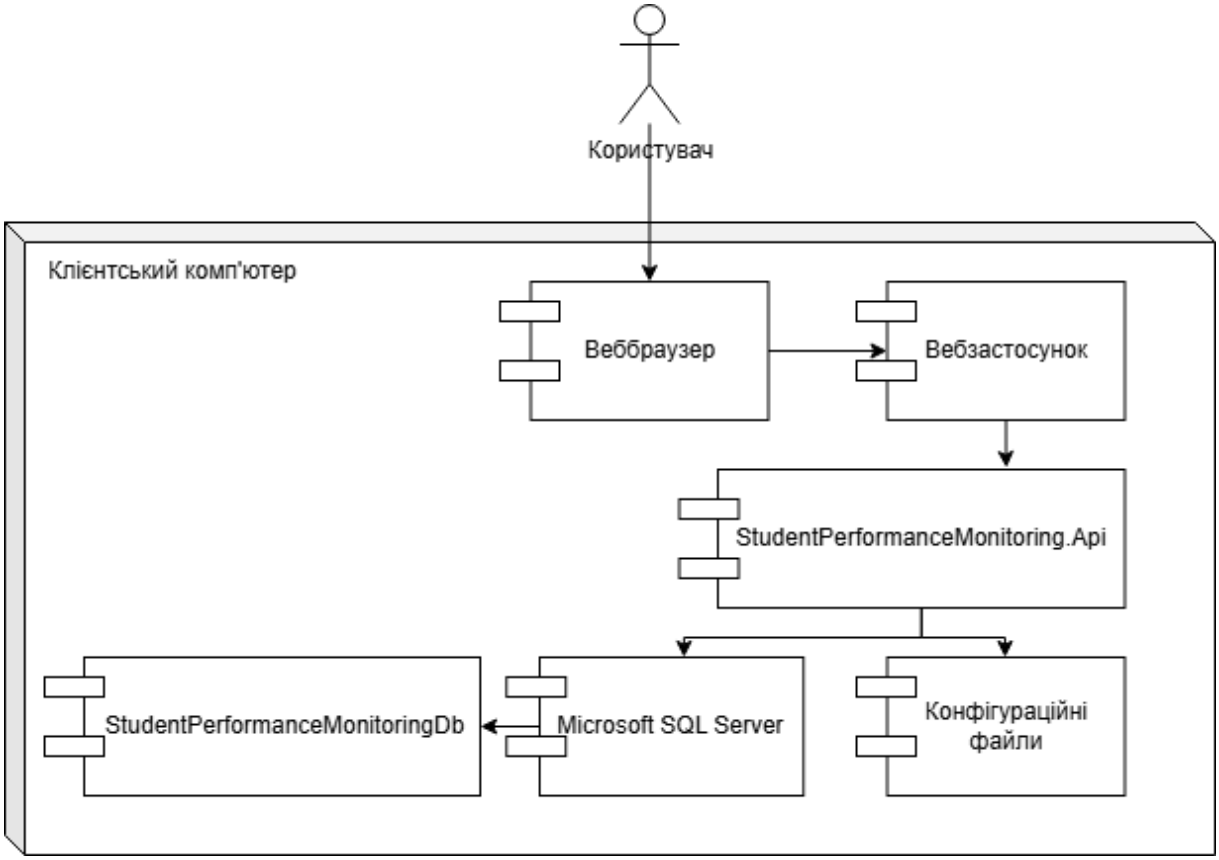


Рис. 4.3 Діаграма локального розгортання

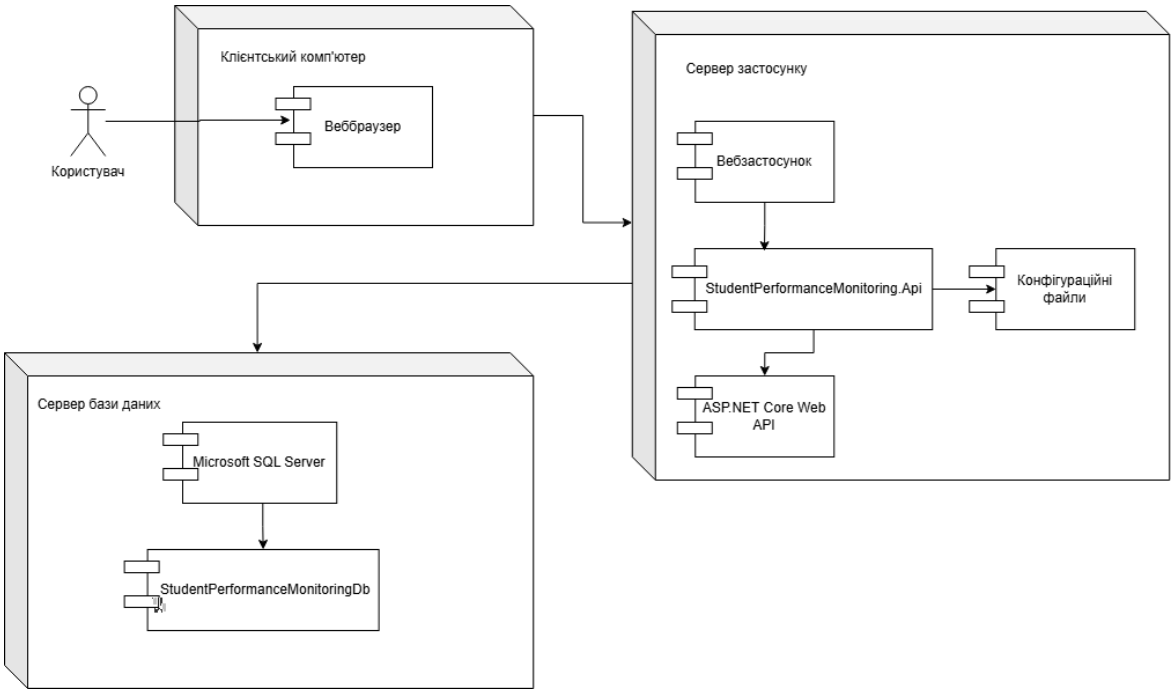


Рис. 4.4 Діаграма віддаленого розгортання

Одним із важливих компонентів інсталяційного пакету є файл конфігурації підключення до бази даних. У даному файлі містяться параметри підключення до Microsoft SQL Server, включаючи адресу сервера, назву бази даних та параметри авторизації.

Використання окремого конфігураційного файлу дозволяє змінювати параметри підключення без необхідності повторної компіляції програмного забезпечення. Це значно спрощує процес налаштування системи в різних умовах експлуатації.

Для забезпечення коректної роботи системи до пакету розгортання також можуть входити SQL-скрипти або механізми міграцій, необхідні для створення чи оновлення структури бази даних. Це дозволяє швидко підготувати інформаційну базу до роботи в новому середовищі.

Для спрощення розгортання програмного забезпечення використовується послідовність публікації серверної частини, збірки клієнтської частини та налаштування параметрів підключення до бази даних. Такий підхід дозволяє адаптувати систему до різних умов експлуатації.

Після завершення процесу розгортання система створює необхідну конфігурацію, підключається до бази даних та стає доступною користувачам через веббраузер. За потреби також перевіряється наявність середовища виконання .NET та доступність Microsoft SQL Server.

Окрему увагу під час формування пакету розгортання було приділено забезпеченню безпеки інформації. Для цього у системі передбачено механізми авторизації користувачів, захисту конфігураційних файлів та розмежування прав доступу.

Під час розробки інсталяційного пакету також враховувалася можливість подальшого оновлення програмного забезпечення. Архітектура системи дозволяє додавати нові модулі та оновлювати програмні компоненти без необхідності повного перевстановлення системи.

Перед оновленням програмного забезпечення доцільно зберігати поточні конфігураційні файли та, за потреби, виконувати резервне копіювання бази даних штатними засобами Microsoft SQL Server. Це дозволяє зменшити ризики втрати даних у випадку помилок під час переходу на нову версію системи.

Після завершення розгортання користувач отримує готову до роботи інформаційну систему, яка забезпечує доступ до основних функціональних можливостей через вебінтерфейс. Усі компоненти системи інтегруються між собою та забезпечують стабільну взаємодію з базою даних.

Таким чином, правильно сформований пакет розгортання забезпечує швидке впровадження веборієнтованої інформаційної системи моніторингу успішності учнів, спрощує процес налаштування програмного забезпечення та забезпечує стабільну роботу системи в умовах експлуатації.

Після завершення розробки інформаційної системи важливим етапом стало забезпечення можливості її практичної демонстрації та віддаленого використання. Оскільки система є веборієнтованою та складається з клієнтської частини, серверної логіки і бази даних, для перевірки її

працездатності в умовах, наближених до реального використання, було організовано демонстраційне розгортання із зовнішнім доступом через мережу Інтернет.

У межах виконаної роботи серверна частина застосунку запускалася локально на комп’ютері розробника, де також знаходилася база даних. Для надання доступу до системи з інших пристроїв було використано сервіс Cloudflare Tunnel. Застосування цього інструмента дозволило створити зовнішнє вебпосилання, через яке користувачі могли відкривати систему в браузері без необхідності придбання окремого віддаленого сервера або оренди платного хостингу. Такий підхід є особливо доцільним для демонстраційної експлуатації, тестування та представлення результатів дипломного проєкту.

Практична цінність такого способу розгортання полягає в тому, що він дає змогу перевірити роботу системи не лише в локальному середовищі, а й у режимі реального віддаленого доступу. Під час тестування було підтверджено можливість відкриття вебзастосунку з інших пристроїв, виконання входу до системи під різними ролями, перегляду наявних даних, а також внесення змін із подальшим збереженням результатів у базі даних. Це означає, що система працює не лише як локальний програмний макет, а як повноцінний вебзастосунок із доступом через мережу.

Важливо зазначити, що в такій схемі клієнтська частина системи відображається в браузері користувача, тоді як обробка запитів, авторизація, бізнес-логіка та робота з базою даних виконуються на серверній частині. Завдяки цьому всі зміни, внесені користувачем, одразу зберігаються в базі даних і можуть бути переглянуті з іншого пристрою. Така організація повністю відповідає клієнт-серверній архітектурі, яка була обрана для розроблюваної інформаційної системи.

Незважаючи на те, що використаний підхід є насамперед демонстраційним, він дозволив підтвердити можливість реального

мережевого доступу до системи. У разі подальшого розвитку програмного продукту аналогічна система може бути перенесена на окремий сервер або хмарну платформу для постійного використання в освітньому закладі. Отже, виконане розгортання можна розглядати як проміжний етап між локальною розробкою та повноцінним продуктивним впровадженням.

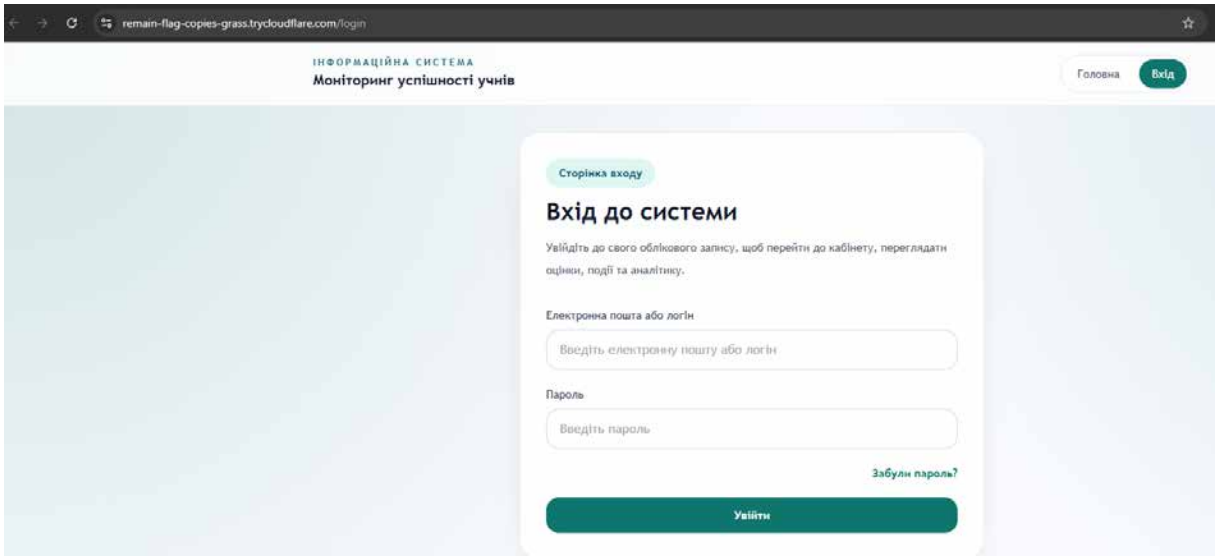


Рис. 4.5 Приклад віддаленого доступу до інформаційної системи через посилання

Таким чином, організація віддаленого доступу до системи дала змогу перевірити її працездатність у реальних умовах використання, продемонструвати коректність взаємодії клієнтської і серверної частин, а також підтвердити можливість використання інформаційної системи з різних пристроїв через браузер.

ВИСНОВКИ

У результаті виконання бакалаврської кваліфікаційної роботи було розроблено інформаційну систему моніторингу успішності учнів, призначену для автоматизації процесів обліку, контролю та аналізу результатів навчальної діяльності у закладах загальної середньої освіти.

Під час виконання роботи було проведено аналіз предметної області та досліджено особливості організації процесу контролю успішності учнів у сучасних навчальних закладах. У ході аналізу було встановлено, що традиційні методи ведення документації, які базуються на використанні паперових журналів або окремих електронних таблиць, мають ряд суттєвих недоліків. До основних проблем належать складність обробки великої кількості інформації, значні витрати часу на формування звітності, відсутність централізованого зберігання даних та обмежені можливості аналітики результатів навчання.

У процесі виконання роботи було досліджено існуючі інформаційні системи та платформи, які використовуються для автоматизації освітнього процесу. Проведений аналіз показав, що сучасні системи мають широкий функціонал, однак значна частина таких рішень є складною для використання у звичайних закладах загальної середньої освіти або містить надлишкові можливості. Це підтвердило доцільність створення власної інформаційної системи, орієнтованої на простоту використання, зручність інтерфейсу та ефективний контроль результатів навчання.

У ході проєктування було виконано моделювання предметної області та визначено основні сутності інформаційної системи, серед яких учні, вчителі, батьки, класи, предмети та оцінки. Для опису структури системи та взаємозв’язків між її компонентами були використані UML-діаграми та ER-моделі бази даних.

На етапі проєктування інформаційного забезпечення було розроблено логічну модель бази даних та створено структуру інформаційної бази у середовищі Microsoft SQL Server. Під час реалізації бази даних було забезпечено цілісність інформації, реалізовано взаємозв’язки між таблицями та створено механізми захисту даних.

Для розробки прикладного програмного забезпечення було використано мову програмування C#, платформу .NET 8, технологію ASP.NET Core Web API, а також React, TypeScript і Vite для клієнтської частини. Обраний інструментарій дозволив реалізувати зручний вебінтерфейс користувача та забезпечити ефективну взаємодію програмного забезпечення з базою даних.

У процесі програмної реалізації було створено основні функціональні модулі системи, серед яких:

– модуль авторизації користувачів;
– модуль електронного журналу;
– модуль роботи з оцінками;
– модуль статистики успішності;
– модуль формування звітів;
– модуль шкільних подій;
– модуль адміністрування системи.

Розроблена система забезпечує можливість автоматичного збереження інформації про результати навчання, формування статистики успішності та створення звітів для учнів, учителів і адміністрації навчального закладу.

Під час виконання роботи також було проведено тестування інформаційної системи. У процесі тестування перевірялася правильність роботи програмних модулів, взаємодія з базою даних, коректність обробки інформації та стабільність функціонування програмного забезпечення. Результати тестування підтвердили правильність роботи системи та

відповідність реалізованого програмного забезпечення поставленим функціональним вимогам.

Окрему увагу було приділено питанням інформаційної безпеки та захисту персональних даних. У системі реалізовано механізми авторизації користувачів, розмежування прав доступу, хешування паролів і контроль активності облікових записів.

Практичне значення отриманих результатів полягає у можливості використання розробленої інформаційної системи у закладах загальної середньої освіти для автоматизації процесу моніторингу успішності учнів. Використання системи дозволяє підвищити ефективність роботи педагогічного персоналу, спростити процес ведення навчальної документації та забезпечити швидкий доступ до актуальної інформації про результати навчання.

Розроблена система може бути використана як основа для подальшого розвитку та розширення функціональних можливостей програмного забезпечення. У майбутньому система може бути доповнена модулями дистанційного навчання, електронних домашніх завдань, мобільним застосунком або інтеграцією з іншими освітніми платформами.

Таким чином, поставлена мета бакалаврської кваліфікаційної роботи була досягнута, а всі поставлені завдання виконані у повному обсязі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Морзе Н. В., Вембер В. П. Інформаційні системи та технології в освіті. – Київ : Видавнича група BHV, 2018. – 384 с.
2. Биков В. Ю. Цифрова трансформація освіти і науки: сучасні виклики та перспективи. – Київ : Інститут цифровізації освіти НАПН України, 2020. – 172 с.
3. Гуржій А. М., Лапінський В. В. Інформаційні технології в освіті. – Київ : Освіта України, 2019. – 256 с.
4. Трофименко О. Г. Проєктування інформаційних систем. – Одеса : Фенікс, 2019. – 268 с.
5. Литвин В. В. Проєктування та розробка інформаційних систем. – Львів : Новий Світ-2000, 2021. – 416 с.
6. Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. – 3rd ed. – Boston : Addison-Wesley, 2004. – 208 p.
7. Pressman R. Software Engineering: A Practitioner’s Approach. – 8th ed. – McGraw-Hill Education, 2015. – 976 p.
8. Sommerville I. Software Engineering. – 10th ed. – Pearson Education, 2016. – 816 p.
9. Руденко В. Д. Основи програмування мовою C#. – Київ : Ліра-К, 2020. – 412 с.
10. Microsoft SQL Server Documentation. – [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/sql/sql-server/
11. Microsoft .NET Documentation. – [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/dotnet/
12. Human – електронний журнал та система управління навчанням. – [Електронний ресурс]. – Режим доступу: https://www.human.ua/
13. Google Classroom Help Center. – [Електронний ресурс]. – Режим доступу: https://support.google.com/edu/classroom

14. Moodle Documentation. – [Електронний ресурс]. – Режим доступу: https://docs.moodle.org/
15. React Documentation. – [Електронний ресурс]. – Режим доступу: https://react.dev/
16. TypeScript Documentation. – [Електронний ресурс]. – Режим доступу: https://www.typescriptlang.org/docs/
17. Vite Documentation. – [Електронний ресурс]. – Режим доступу: https://vite.dev/guide/
18. ASP.NET Core Documentation. – [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/aspnet/core
19. Entity Framework Core Documentation. – [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/ef/core/
20. Типи баз даних: особливості, відмінності та приклади. – [Електронний ресурс]. – Режим доступу: https://dou.ua/lenta/articles/types-of-databases/

Додаток А

РЕАЛІЗАЦІЯ МОДУЛЯ ЕЛЕКТРОННОГО ЖУРНАЛУ ТА РОЛЬОВОГО ДОСТУПУ ДО ОЦІНОК

Сторінок – 5

2026

```
     using StudentPerformanceMonitoring.Application.DTOs.Grades;
     using StudentPerformanceMonitoring.Application.DTOs.Semesters;
     using StudentPerformanceMonitoring.Application.Exceptions;
     using
StudentPerformanceMonitoring.Application.Interfaces.Repositories;
     using
StudentPerformanceMonitoring.Application.Interfaces.Services;
     using StudentPerformanceMonitoring.Application.Models.Grades;
     using StudentPerformanceMonitoring.Domain.Constants;
     using StudentPerformanceMonitoring.Domain.Entities;

     namespace StudentPerformanceMonitoring.Application.Services;

     public sealed class GradeService(IGradeRepository repository) :
IGradeService
     {
         public    async    Task<IReadOnlyCollection<GradeListDto>>
GetGradesAsync(
             GradeFilterQueryDto query,
             Guid currentUserId,
             string currentRole,
             CancellationToken cancellationToken = default)
         {
             ValidateDateRange(query.DateFrom, query.DateTo);

             var filter = new GradeFilterParameters
             {
                 ClassId = query.ClassId,
                 SubjectId = query.SubjectId,
                 TeacherId = query.TeacherId,
                 SemesterId = query.SemesterId,
                 DateFrom = query.DateFrom,
                 DateTo = query.DateTo
             };

             switch (currentRole)
             {
                 case SystemRoleNames.Leader:
                     if (query.StudentId.HasValue)
                     {
                         filter.StudentIds                          =
[query.StudentId.Value];
                     }

                     break;

                 case SystemRoleNames.Teacher:
                 {
                     var          teacher          =          await
repository.GetTeacherByUserIdAsync(currentUserId, cancellationToken)
                         ??                throw                new
NotFoundRequestException("Профіль учителя не знайдено.");

                     if         (query.TeacherId.HasValue         &&
query.TeacherId.Value != teacher.Id)
                     {
```

```
                        throw new ForbiddenRequestException("Ви
можете переглядати лише власний журнал оцінок.");
                    }

                    filter.TeacherId = teacher.Id;

                    if (query.StudentId.HasValue)
                    {
                        filter.StudentIds                        =
[query.StudentId.Value];
                    }

                    break;
                }

                case SystemRoleNames.Student:
                {
                    var          student          =          await
repository.GetStudentByUserIdAsync(currentUserId, cancellationToken)
                        ??                throw                new
NotFoundRequestException("Профіль учня не знайдено.");

                    if         (query.StudentId.HasValue         &&
query.StudentId.Value != student.Id)
                    {
                        throw new ForbiddenRequestException("Ви
можете переглядати лише власні оцінки.");
                    }

                    if          (query.ClassId.HasValue          &&
query.ClassId.Value != student.SchoolClassId)
                    {
                        throw new ForbiddenRequestException("Ви
можете переглядати лише дані свого класу.");
                    }

                    filter.StudentIds = [student.Id];
                    break;
                }

                case SystemRoleNames.Parent:
                {
                    var          childIds          =          await
repository.GetParentChildrenStudentIdsAsync(currentUserId,
cancellationToken);

                    if (childIds.Count == 0)
                    {
                        return Array.Empty<GradeListDto>();
                    }

                    if (query.StudentId.HasValue)
                    {
                        if
(!childIds.Contains(query.StudentId.Value))
                        {
```

```
                            throw new ForbiddenRequestException("Ви
можете переглядати оцінки лише своєї дитини.");
                        }

                        filter.StudentIds                           =
[query.StudentId.Value];
                    }
                    else
                    {
                        filter.StudentIds = childIds;
                    }

                    break;
                }

                default:
                    throw  new  UnauthorizedRequestException("Сесію
завершено. Увійдіть повторно.");
            }

            var  grades  =  await  repository.GetGradesAsync(filter,
cancellationToken);

            return grades
                .Select(MapGrade)
                .ToArray();
        }

        public    async    Task<IReadOnlyCollection<GradeListDto>>
GetStudentGradesAsync(
            Guid studentId,
            GradeFilterQueryDto query,
            Guid currentUserId,
            string currentRole,
            CancellationToken cancellationToken = default)
        {
            ValidateDateRange(query.DateFrom, query.DateTo);

            var filter = new GradeFilterParameters
            {
                ClassId = query.ClassId,
                SubjectId = query.SubjectId,
                TeacherId = query.TeacherId,
                SemesterId = query.SemesterId,
                DateFrom = query.DateFrom,
                DateTo = query.DateTo,
                StudentIds = [studentId]
            };

            switch (currentRole)
            {
                case SystemRoleNames.Leader:
                    break;

                case SystemRoleNames.Teacher:
                {
```

```
                    var teacher = await
repository.GetTeacherByUserIdAsync(currentUserId, cancellationToken)
                        ?? throw new
NotFoundRequestException("Профіль учителя не знайдено.");
                    var student = await
repository.GetStudentByIdAsync(studentId, cancellationToken)
                        ?? throw new
NotFoundRequestException("Профіль учня не знайдено.");

                    if (!await
repository.TeacherHasClassAccessAsync(
                            currentUserId,
                            student.SchoolClassId,
                            cancellationToken))
                    {
                        throw new ForbiddenRequestException("Ви
можете переглядати оцінки лише своїх учнів.");
                    }

                    if (query.TeacherId.HasValue &&
query.TeacherId.Value != teacher.Id)
                    {
                        throw new ForbiddenRequestException("Ви
можете переглядати лише власний журнал оцінок.");
                    }

                    filter.TeacherId = teacher.Id;
                    break;
                }

                case SystemRoleNames.Student:
                {
                    var student = await
repository.GetStudentByUserIdAsync(currentUserId, cancellationToken)
                        ?? throw new
NotFoundRequestException("Профіль учня не знайдено.");

                    if (student.Id != studentId)
                    {
                        throw new ForbiddenRequestException("Ви
можете переглядати лише власні оцінки.");
                    }

                    filter.StudentIds = [student.Id];
                    break;
                }

                case SystemRoleNames.Parent:
                {
                    var childIds = await
repository.GetParentChildrenStudentIdsAsync(currentUserId,
cancellationToken);

                    if (!childIds.Contains(studentId))
                    {
```

```
                        throw new ForbiddenRequestException("Ви
можете переглядати оцінки лише своєї дитини.");
                    }

                    break;
                }

                default:
                    throw new UnauthorizedRequestException("Сесію
завершено. Увійдіть повторно.");
            }

            var grades = await repository.GetGradesAsync(filter,
cancellationToken);

            return grades
                .Select(MapGrade)
                .ToArray();
        }
    }
```

Додаток Б

РЕАЛІЗАЦІЯ МОДУЛЯ АНАЛІТИКИ УСПІШНОСТІ У РОЛІ КЕРІВНИКА

Сторінок – 7

2026

```
using StudentPerformanceMonitoring.Application.DTOs.Grades;
using
StudentPerformanceMonitoring.Application.DTOs.LeaderCabinet;
using
StudentPerformanceMonitoring.Application.DTOs.StudentCabinet;
using StudentPerformanceMonitoring.Application.Exceptions;
using
StudentPerformanceMonitoring.Application.Interfaces.Repositories;
using
StudentPerformanceMonitoring.Application.Interfaces.Services;
using StudentPerformanceMonitoring.Domain.Entities;

namespace StudentPerformanceMonitoring.Application.Services;

public sealed class
LeaderCabinetService(ILeaderCabinetRepository repository) :
ILeaderCabinetService
{
    private const decimal PerformanceShiftThreshold = 1.0m;
    private const string ImprovementStatus = "Покращення";
    private const string StableStatus = "Стабільні";
    private const string WorseningStatus = "Погіршення";
    private const string NoBaselineStatus = "Немає бази для
порівняння";
    private const string AllMode = "all";
    private const string WorseningMode = "worsening";
    private const string ImprovementMode = "improvement";
    private const string StableMode = "stable";
    private const string AcademicYearMode = "academicYear";

    public async Task<LeaderSummaryStatisticsDto>
GetSummaryAsync(
        LeaderCabinetFilterQueryDto query,
        CancellationToken cancellationToken = default)
    {
        var analytics = await BuildAnalyticsContextAsync(query,
cancellationToken);
        var overviewCounts = await
repository.GetOverviewCountsAsync(query.ClassId, cancellationToken);

        return new LeaderSummaryStatisticsDto
        {
            ClassCount = overviewCounts.ClassCount,
            StudentCount = overviewCounts.StudentCount,
            TeacherCount = overviewCounts.TeacherCount,
            OverallAverageGrade = analytics.CurrentGrades.Count
== 0
                ? 0
                :
decimal.Round(analytics.CurrentGrades.Average(grade => grade.Value),
2)
        };
    }
```

```
        public                                                  async
Task<IReadOnlyCollection<LeaderClassPerformanceDto>>
GetClassPerformanceAsync(
            LeaderCabinetFilterQueryDto query,
            CancellationToken cancellationToken = default)
        {
            var analytics = await BuildAnalyticsContextAsync(query,
cancellationToken);

            return analytics.CurrentGrades
                .GroupBy(grade => new
                {
                    grade.Student.SchoolClassId,
                    grade.Student.SchoolClass.Name,
                    grade.Student.SchoolClass.AcademicYear
                })
                .Select(group => new LeaderClassPerformanceDto
                {
                    SchoolClassId = group.Key.SchoolClassId,
                    ClassName = group.Key.Name,
                    AcademicYear = group.Key.AcademicYear,
                    StudentCount     =     group.Select(item     =>
item.StudentId).Distinct().Count(),
                    GradeCount = group.Count(),
                    AverageGrade = decimal.Round(group.Average(item
=> item.Value), 2)
                })
                .OrderBy(item => item.AcademicYear)
                .ThenBy(item => item.ClassName)
                .ToArray();
        }

        public                                                  async
Task<IReadOnlyCollection<LeaderSubjectPerformanceDto>>
GetSubjectPerformanceAsync(
            LeaderCabinetFilterQueryDto query,
            CancellationToken cancellationToken = default)
        {
            var analytics = await BuildAnalyticsContextAsync(query,
cancellationToken);

            return analytics.CurrentGrades
                .GroupBy(grade    =>    new    {    grade.SubjectId,
grade.Subject.Name })
                .Select(group => new LeaderSubjectPerformanceDto
                {
                    SubjectId = group.Key.SubjectId,
                    SubjectName = group.Key.Name,
                    StudentCount     =     group.Select(item     =>
item.StudentId).Distinct().Count(),
                    GradeCount = group.Count(),
                    AverageGrade = decimal.Round(group.Average(item
=> item.Value), 2)
                })
                .OrderBy(item => item.SubjectName)
                .ToArray();
```

```
        }

        public                                                  async
Task<IReadOnlyCollection<LeaderWorseningStudentDto>>
GetWorseningStudentsAsync(
            LeaderCabinetFilterQueryDto query,
            CancellationToken cancellationToken = default)
        {
            var analytics = await BuildAnalyticsContextAsync(query,
cancellationToken);
            var                  performanceMode                  =
NormalizePerformanceMode(query.PerformanceMode);

            return analytics.CurrentGrades
                .GroupBy(grade => new
                {
                    grade.StudentId,
                    FullName = FormatFullName(grade.Student.User),
                    grade.Student.SchoolClass.Name,
                    grade.Student.SchoolClass.AcademicYear
                })
                .Select(group =>
                {
                    var              currentAverage              =
decimal.Round(group.Average(item => item.Value), 2);
                    var previousValues = analytics.BaselineGrades
                        .Where(item     =>     item.StudentId     ==
group.Key.StudentId)
                        .Select(item => item.Value)
                        .ToArray();

                    decimal? previousAverage = previousValues.Length
== 0
                        ? null
                        :   decimal.Round(previousValues.Average(),
2);
                    decimal? difference = previousAverage.HasValue
                        ?       decimal.Round(currentAverage      -
previousAverage.Value, 2)
                        : null;
                    var status = ResolveStatus(difference);

                    if             (!MatchesPerformanceMode(status,
performanceMode))
                    {
                        return null;
                    }

                    return new LeaderWorseningStudentDto
                    {
                        StudentId = group.Key.StudentId,
                        FullName = group.Key.FullName,
                        ClassName = group.Key.Name,
                        AcademicYear = group.Key.AcademicYear,
                        GradeCount = group.Count(),
                        CurrentAverage = currentAverage,
```

```
                        PreviousAverage = previousAverage,
                        Difference = difference,
                        Status = status
                    };
                })
                .Where(item => item is not null)
                .Select(item => item!)
                .OrderBy(item => GetStatusOrder(item.Status))
                .ThenBy(item   =>   GetDifferenceOrder(item.Status,
item.Difference))
                .ThenBy(item => item.FullName)
                .ToArray();
        }

        public          async          Task<LeaderStudentDetailDto>
GetStudentDetailAsync(
            Guid studentId,
            LeaderCabinetFilterQueryDto query,
            CancellationToken cancellationToken = default)
        {
            var analytics = await BuildAnalyticsContextAsync(query,
cancellationToken);
            var             student             =             await
repository.GetStudentByIdAsync(studentId, cancellationToken)
                ?? throw new NotFoundRequestException("Профіль учня
не знайдено.");

            var studentCurrentGrades = analytics.CurrentGrades
                .Where(grade => grade.StudentId == studentId)
                .ToArray();
            var studentBaselineGrades = Array.Empty<Grade>();

            return new LeaderStudentDetailDto
            {
                StudentId = student.Id,
                FullName = FormatFullName(student.User),
                Email = student.User.Email,
                ClassName = student.SchoolClass.Name,
                AcademicYear = student.SchoolClass.AcademicYear,
                TotalGrades = studentCurrentGrades.Length,
                AverageGrade = studentCurrentGrades.Length == 0
                    ? 0
                    :
decimal.Round(studentCurrentGrades.Average(grade => grade.Value), 2),
                SubjectPerformance                                  =
BuildStudentSubjectPerformance(studentCurrentGrades,
studentBaselineGrades),
                Trend = BuildTrend(studentCurrentGrades),
                RecentGrades = studentCurrentGrades
                    .OrderByDescending(grade => grade.GradeDate)
                    .ThenByDescending(grade => grade.CreatedAt)
                    .Take(5)
                    .Select(MapGrade)
                    .ToArray()
            };
        }
```

```
        private async Task<AnalyticsContext>
BuildAnalyticsContextAsync(
            LeaderCabinetFilterQueryDto query,
            CancellationToken cancellationToken)
        {
            ValidateDateRange(query.DateFrom, query.DateTo);

            var structuralGrades = await repository.GetGradesAsync(
                query.ClassId,
                query.SubjectId,
                cancellationToken);
            var semesters = await
repository.GetSemestersAsync(cancellationToken);
            var academicYearPeriod =
ResolveAcademicYearPeriod(semesters, query);
            var currentGrades =
FilterCurrentGrades(structuralGrades, semesters, query,
academicYearPeriod);
            var baselineGrades = BuildBaselineGrades(
                structuralGrades,
                semesters,
                query,
                currentGrades,
                academicYearPeriod);

            return new AnalyticsContext(structuralGrades,
currentGrades, baselineGrades);
        }

        private static string ResolveStatus(decimal? difference)
        {
            if (!difference.HasValue)
            {
                return NoBaselineStatus;
            }

            if (difference.Value >= PerformanceShiftThreshold)
            {
                return ImprovementStatus;
            }

            if (difference.Value <= -PerformanceShiftThreshold)
            {
                return WorseningStatus;
            }

            return StableStatus;
        }

        private static string NormalizePerformanceMode(string?
performanceMode)
        {
            if (string.IsNullOrWhiteSpace(performanceMode))
            {
                return WorseningMode;
```

```
            }

            return performanceMode.Trim().ToLowerInvariant() switch
            {
                ImprovementMode => ImprovementMode,
                StableMode => StableMode,
                AllMode => AllMode,
                _ => WorseningMode
            };
        }

        private static bool MatchesPerformanceMode(string status,
string performanceMode)
        {
            if (status == NoBaselineStatus)
            {
                return performanceMode == AllMode;
            }

            return performanceMode switch
            {
                AllMode   =>   status   is   WorseningStatus   or
ImprovementStatus or StableStatus,
                ImprovementMode => status == ImprovementStatus,
                StableMode => status == StableStatus,
                _ => status == WorseningStatus
            };
        }

        private static int GetStatusOrder(string status)
        {
            return status switch
            {
                WorseningStatus => 0,
                StableStatus => 1,
                ImprovementStatus => 2,
                _ => 3
            };
        }

        private  static  decimal  GetDifferenceOrder(string  status,
decimal? difference)
        {
            if (!difference.HasValue)
            {
                return decimal.MaxValue;
            }

            return status switch
            {
                StableStatus => Math.Abs(difference.Value),
                ImprovementStatus => -difference.Value,
                _ => difference.Value
            };
        }
```

```
        private static void ValidateDateRange(DateOnly? dateFrom,
DateOnly? dateTo)
        {
            if (dateFrom.HasValue && dateTo.HasValue &&
dateFrom.Value > dateTo.Value)
            {
                throw new BadRequestException("Початкова дата не
може бути пізнішою за кінцеву.");
            }
        }

        private static GradeListDto MapGrade(Grade grade)
        {
            return new GradeListDto
            {
                Id = grade.Id,
                StudentId = grade.StudentId,
                StudentFullName =
FormatFullName(grade.Student.User),
                TeacherId = grade.TeacherId,
                TeacherFullName =
FormatFullName(grade.Teacher.User),
                SubjectId = grade.SubjectId,
                SubjectName = grade.Subject.Name,
                SchoolClassId = grade.Student.SchoolClassId,
                SchoolClassName = grade.Student.SchoolClass.Name,
                AcademicYear =
grade.Student.SchoolClass.AcademicYear,
                SemesterId = grade.SemesterId,
                SemesterName = grade.Semester.Name,
                Value = grade.Value,
                GradeDate = grade.GradeDate,
                Comment = grade.Comment,
                CreatedAt = grade.CreatedAt
            };
        }

        private static string FormatFullName(User user)
        {
            return string.Join(
                " ",
                new[] { user.LastName, user.FirstName,
user.MiddleName }
                    .Where(part =>
!string.IsNullOrWhiteSpace(part)));
        }

        private sealed record AnalyticsContext(
            IReadOnlyCollection<Grade> StructuralGrades,
            IReadOnlyCollection<Grade> CurrentGrades,
            IReadOnlyCollection<Grade> BaselineGrades);
    }
```

ДОДАТОК В

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

Декларація про академічну доброчесність та використання ШІ

ДЕКЛАРАЦІЯ ЗДОБУВАЧА

Я, Долобашко Данііл Олександрович, здобувач(ка) НУБіП України, усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що:

1. Моя бакалаврська кваліфікаційна робота (проєкт) на тему «Інформаційна система моніторингу успішності учнів» 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.
    - [X] інструмент ШІ ChatGPT був використаний для:
        - [ ] перекладу іншомовних джерел;
        - [X] стилістичного редагування та перевірки граматики;
        - [X] 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.

«___» _____ 2026 р. _____ **Данііл ДОЛОБАШКО**
(підпис)