НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

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

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: « Інформаційна система обліку та аналізу діяльності студентських наукових гуртків**»**
Спеціальність «122 Комп'ютерні науки»
Освітньо-професійна програма «Комп'ютерні науки»

Гарант освітньої програми
к.т.н., доцент _____ **Роман РУДЕНСЬКИЙ**
(підпис)

Керівник бакалаврської кваліфікаційної роботи
ст.викладач _____ **Максим Мокрієв**
(підпис)

Виконав _____ Липницький Ярослав Андрійович
(підпис)

Київ-2026

НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ЗАТВЕРДЖУЮЗ

авідувач кафедри інформаційних систем і технологій

к.е.н., доцент _____ Михайло ШВИДЕНКО

(підпис)

«___» _____ 2026 р.

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

_____

Липницькому Ярославу Андрійовичу

Спеціальність «122 Комп'ютерні науки»

Освітньо-професійна програма «Комп'ютерні науки»

1. Тема бакалаврської кваліфікаційної роботи
«Інформаційна система обліку та аналізу діяльності студентських наукових гуртків» затверджена наказом від 06.01.2026 №46 «С».
2. Термін подання завершеної роботи на кафедру 2026.05.25
3. Вихідні дані до бакалаврської кваліфікаційної роботи: Аналіз предметної області студентських наукових гуртків: Відомості про студентів, керівників гуртків, студентські наукові гуртки, склад учасників, заявки на вступ, наукові заходи, відвідуваність, результати діяльності студентів
4. **Перелік питань, що підлягають дослідженню:**
   - Аналіз предметної області та визначення функціональних вимог до інформаційної системи обліку та аналізу діяльності студентських наукових гуртків.
   - Проєктування структури інформаційної системи та моделі даних.
   - Розробка програмного забезпечення для автоматизації обліку студентів, наукових гуртків, заявок, відвідуваності, заходів і результатів діяльності.
   - Реалізація механізмів формування звітності та аналітичної інформації для адміністратора, керівника гуртка та деканату.
   - Тестування та оцінка ефективності функціонування розробленої системи.

Дата видачі завдання «_06__» Січня_____ 2026 р.

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

Завдання прийняв до виконання:_ _____ / Липницький Я.А. /

Студент _____ ____Липницький Я.А. _____

(підпис) (прізвище та ініціали)

Керівник бакалаврської кваліфікаційної роботи __ _Мокрієв М.В,

(підпис) (прізвище та інінціали)

Зміст

ВСТУП.....5

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ.....8

1.1 Аналіз предметної області діяльності студентських наукових гуртків.....8

1.2 Огляд інформаційних джерел та існуючих рішень у сфері обліку наукової діяльності студентів.....10

1.3 Моделювання предметної області інформаційної системи обліку та аналізу діяльності студентських наукових гуртків.....15

1.4 Аналіз вимог програмної системи.....19

1.5 Постановка завдання.....21

2.....ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ 23

2.1 Логічна модель даних.....24

2.2 Вибір системи управління інформаційною базою.....28

2.3 Створення інформаційної бази.....31

3 ПРИКЛАДНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ.....36

3.1 Організаційна структура програмного забезпечення.....37

3.2 Вибір інструментарію для створення ППЗ.....39

3.3 Алгоритмізація та програмування програмних модулів.....43

4 РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ СИСТЕМИ.....58

4.1 Тестування системи.....59

4.2 Вимоги до апаратного та програмного забезпечення.....66

4.3 Склад інсталяційного пакету.....70

ВИСНОВОК.....74

Список використаних джерел.....76

Додаток А – Діаграма прецедентів.....77

Додаток Б – Діаграма діяльності.....78

Додаток В – Функціональна діаграма.....80

Додаток Г – Діаграма послідовності.....80

Додаток Д – Діаграма розгортання.....81

Додаток Е – Логічна модель БД.....82

Додаток Ж – Program.cs.....83

Додаток З – Декларація ШІ.....92

ВСТУП

Особливого значення набуває організація, облік та аналіз наукової діяльності студентів. Однією з ключових форм такої діяльності є студентські наукові гуртки, які забезпечують залучення здобувачів освіти до дослідницької роботи, сприяють формуванню наукового мислення, розвитку професійних компетентностей та підвищенню рівня підготовки майбутніх фахівців. Ефективне функціонування таких гуртків передбачає систематичний облік їх учасників, планування та фіксацію проведених заходів, збереження результатів наукової діяльності, а також оцінювання рівня активності та досягнень студентів.

Зростає потреба у створенні інформаційних систем для автоматизації обліку та аналізу діяльності студентських наукових гуртків. Традиційні підходи, що ґрунтуються на паперовій документації або розрізнених електронних засобах, ускладнюють централізоване зберігання даних, формування звітності та прийняття управлінських рішень.

У зв’язку з цим актуальним є розроблення інформаційної системи, яка забезпечує збір, зберігання, оброблення та аналіз даних про діяльність студентських наукових гуртків. Використання такої системи дозволяє вести облік учасників, заходів і результатів наукової діяльності, формувати аналітичні звіти та підвищувати ефективність організації освітньо-наукового процесу.

Мета роботи полягає у розробленні інформаційної системи обліку та аналізу діяльності студентських наукових гуртків, яка забезпечує автоматизацію процесів ведення обліку, аналітичної обробки даних та формування звітності щодо результатів наукової діяльності студентів.

Для досягнення поставленої мети необхідно вирішити наступні **завдання:**

1. Провести системний аналіз предметної області діяльності студентських наукових гуртків, охарактеризувати існуючі підходи до обліку та аналізу наукової активності студентів і визначити недоліки сучасних рішень.

2. Розробити логічну, інформаційну та архітектурну моделі інформаційної системи обліку та аналізу діяльності студентських наукових гуртків.

3. Формалізувати функціональні, нефункціональні й технічні вимоги до системи.

4. Реалізувати програмні модулі управління користувачами, обліку діяльності гуртків, аналітики та формування звітів.

5. Провести дослідження ефективності функціонування системи та оцінити її вплив на якість обліку й аналізу наукової діяльності студентів.

Об’єктом дослідження є процес організації, обліку та аналізу діяльності студентських наукових гуртків у закладах вищої освіти.

Предметом дослідження є методи, моделі та інформаційно-технологічні засоби створення інформаційних систем, орієнтованих на автоматизацію обліку та аналітичної обробки даних щодо наукової діяльності студентів.

Методи дослідження. У роботі використано комплекс сучасних підходів, зокрема методи системного аналізу, методи об’єктно-орієнтованого проєктування, моделювання інформаційних систем із застосуванням нотації UML, методи аналізу даних та формування звітності. Для реалізації програмної системи застосовано сучасні веб-технології, що забезпечують створення гнучкої, масштабованої та зручної у використанні інформаційної платформи.

Наукова новизна одержаних результатів полягає у розробленні узагальненої моделі інформаційної системи обліку та аналізу діяльності студентських наукових гуртків, яка поєднує механізми централізованого зберігання даних, аналітичної обробки показників наукової активності та формування звітності. Запропонований підхід дозволяє підвищити ефективність

управління науковою діяльністю студентів, забезпечити оперативний доступ до актуальної інформації та покращити якість прийняття управлінських рішень.

Структура роботи. Бакалаврська кваліфікаційна робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел та додатків. У першому розділі виконано системний аналіз предметної області, розглянуто існуючі інформаційні джерела й рішення, сформульовано вимоги до системи та постановку завдання. У другому розділі описано інформаційне забезпечення системи, логічну модель даних, вибір системи управління базою даних і створення інформаційної бази. У третьому розділі розглянуто прикладне програмне забезпечення, його організаційну структуру, інструментарій розробки та алгоритмізацію програмних модулів. У четвертому розділі наведено результати тестування системи, вимоги до апаратного й програмного забезпечення, а також склад інсталяційного пакету.

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз предметної області діяльності студентських наукових гуртків

У сучасних умовах функціонування закладів вищої освіти діяльність студентських наукових гуртків розглядається як важливий елемент освітньо-наукового процесу, спрямований на формування дослідницьких компетентностей, розвиток аналітичного мислення та підготовку студентів до наукової діяльності. Організація роботи гуртків передбачає взаємодію різних учасників, зокрема студентів, керівників гуртків, адміністрації та інформаційних ресурсів університету, що обумовлює наявність складної інформаційної взаємодії та необхідність її формалізації.

На рисунку 1.1 наведено узагальнену модель предметної області, яка відображає основні інформаційні потоки та взаємодію зовнішніх сутностей із системою обліку та аналізу діяльності студентських наукових гуртків.

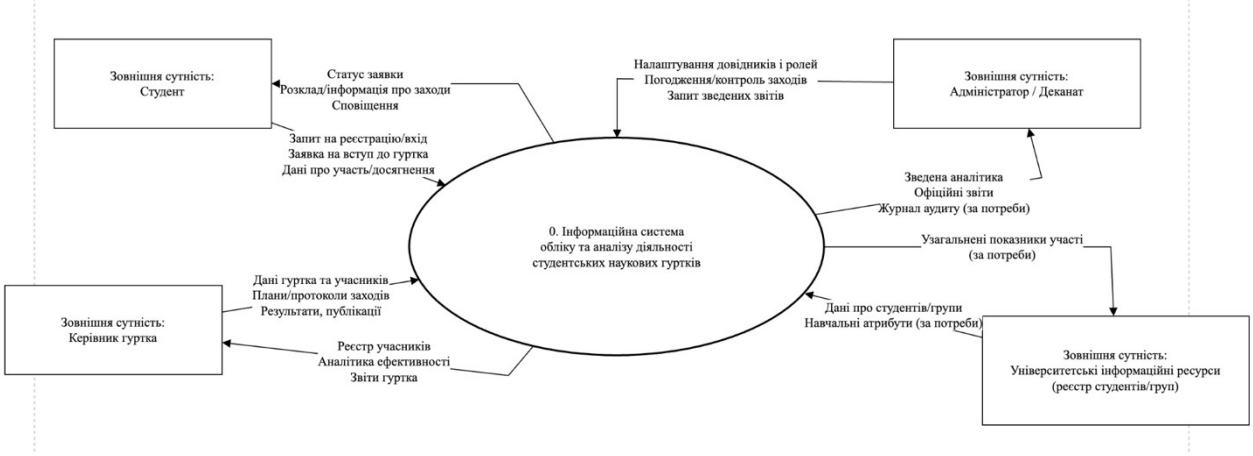


Рисунок 1.1 – Контекстна діаграма інформаційної системи обліку та аналізу діяльності студентських наукових гуртків

Предметна область охоплює процеси реєстрації та обліку учасників гуртків, планування та проведення заходів, фіксації результатів наукової

діяльності, а також формування аналітичних показників ефективності. Взаємодія студентів із системою передбачає подання заявок на участь, отримання інформації про заходи та доступ до результатів власної наукової активності. Керівники гуртків забезпечують введення та актуалізацію даних щодо складу учасників, планів роботи, результатів досліджень і публікацій. Адміністративний персонал здійснює контроль, формування зведених звітів та аналіз ефективності діяльності гуртків, що є основою для прийняття управлінських рішень.

Важливим аспектом предметної області є інтеграція з університетськими інформаційними ресурсами, які забезпечують доступ до актуальних даних про студентів та їх навчальні характеристики. Це дозволяє підвищити достовірність даних та уникнути дублювання інформації. Таким чином, система виступає як централізований інструмент накопичення, оброблення та аналізу інформації про наукову діяльність студентів.

Для формалізації ключових сутностей предметної області та їх функціонального призначення узагальнені характеристики наведено в таблиці 1.1.

Таблиця 1.1 – Основні сутності предметної області та їх характеристика

Сутність	Опис	Основні функції
Студент	Учасник наукового гуртка	Реєстрація, подання заявок, участь у заходах, перегляд результатів
Керівник гуртка	Особа, що організовує діяльність гуртка	Формування складу гуртка, планування заходів, внесення результатів
Адміністратор / деканат	Управлінський персонал	Контроль діяльності, формування звітів, аналітика
Науковий гурток	Організаційна одиниця	Облік учасників, проведення заходів, накопичення результатів
Захід	Подія в межах гуртка	Планування, проведення, фіксація результатів
Інформаційні ресурси університету	Зовнішні системи	Надання даних про студентів і групи

Аналіз предметної області дозволяє встановити, що основними проблемами існуючих підходів є фрагментованість даних, відсутність централізованого обліку, складність формування аналітичної інформації та обмежені можливості оцінювання ефективності діяльності студентських наукових гуртків. Це обумовлює необхідність розроблення інтегрованої інформаційної системи, яка забезпечить автоматизацію процесів обліку, підвищення достовірності даних та підтримку прийняття управлінських рішень на основі аналітичних показників.

1.2 Огляд інформаційних джерел та існуючих рішень у сфері обліку наукової діяльності студентів

Розвиток цифрових технологій у сфері науки й освіти сприяв появі систем для обліку, аналізу та представлення результатів наукової діяльності. Такі платформи накопичують інформацію про публікації, індексують дослідницьку активність і формують аналітичні показники. До найбільш відомих рішень належать ORCID, Google Scholar, Scopus, Web of Science та Publons.

ORCID є глобальним сервісом ідентифікації дослідників, який забезпечує створення унікального профілю науковця та об’єднання інформації про його публікації, проєкти й наукову активність. Вебінтерфейс системи, наведений на рисунку 1.2, дає змогу централізовано управляти науковими даними.

Рисунок 1.2 – Веб-інтерфейс системи ORCID для управління профілем дослідника

Платформа Google Scholar (рисунок 1.3) орієнтована на індексацію наукових публікацій та автоматичне формування бібліометричних показників, зокрема індексу цитування. Система забезпечує швидкий пошук наукових матеріалів, а також надає можливість створення персональних профілів дослідників із відображенням їхньої наукової активності.

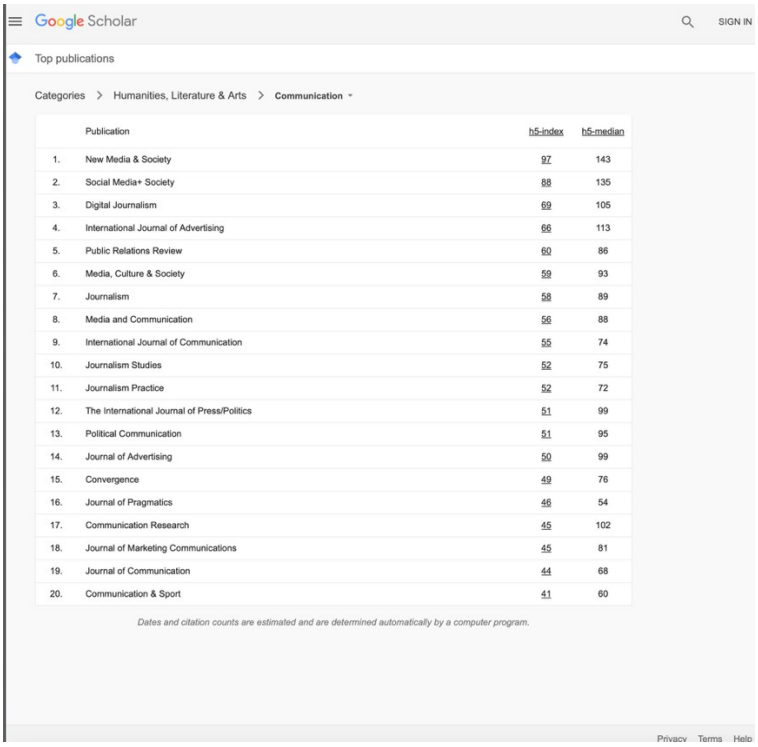


Рисунок 1.3 – Інтерфейс Google Scholar із відображенням наукових публікацій та метрик цитування

Scopus (рисунок 1.4) є однією з найбільших реферативних і аналітичних баз даних, яка надає інструменти для оцінювання наукової діяльності, аналізу публікаційної активності та побудови наукових рейтингів. Система використовує складні алгоритми оброблення даних для формування показників ефективності досліджень.

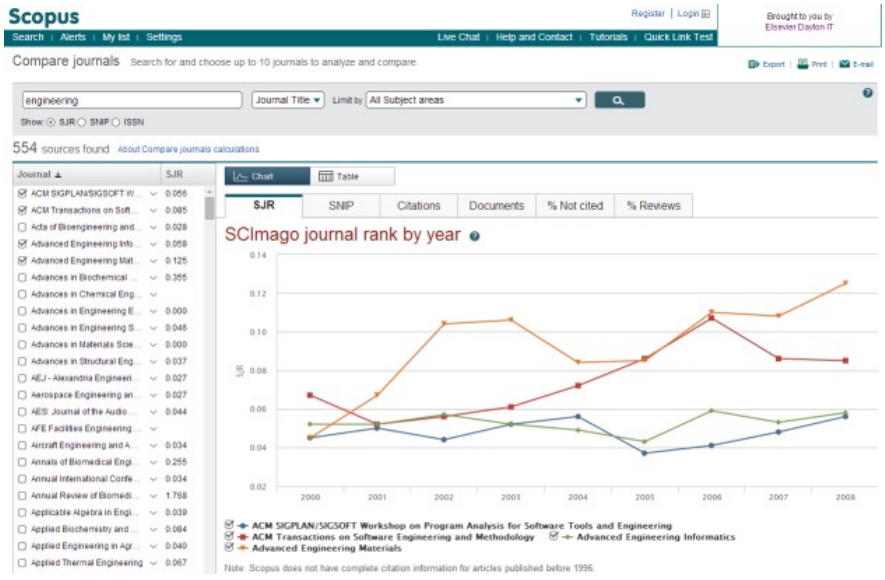


Рисунок 1.4 – Аналітичний інтерфейс системи Scopus для оцінювання наукової діяльності

Платформа Web of Science (рисунок 1.5) забезпечує доступ до наукових публікацій, аналітичних інструментів і бібліометричних показників. Вона дозволяє здійснювати комплексний аналіз наукової діяльності дослідників, організацій і галузей знань, а також формувати звіти для прийняття управлінських рішень.

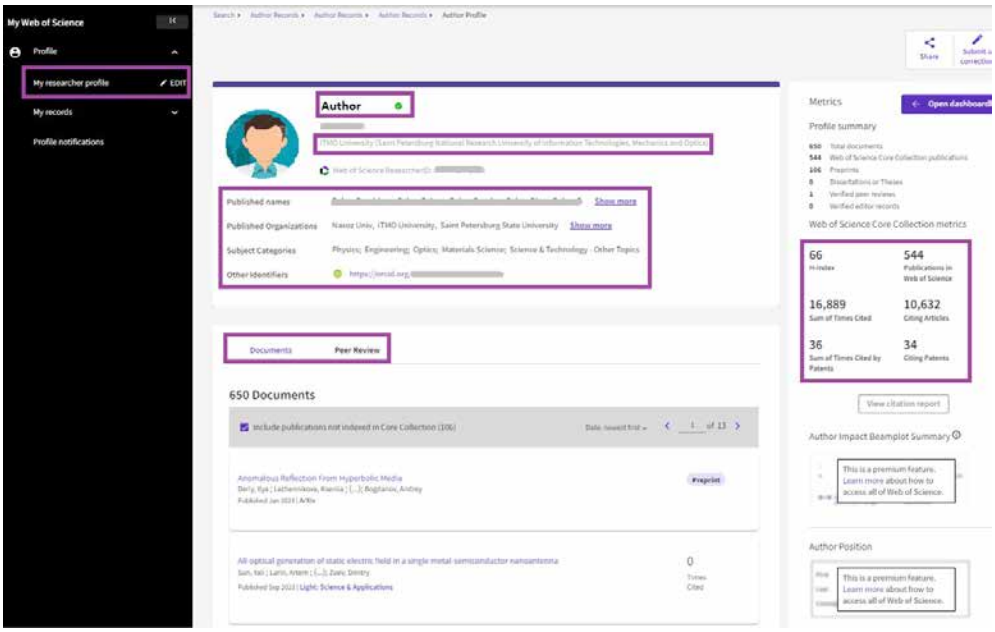


Рисунок 1.5 – Профіль дослідника у системі Web of Science із показниками наукової активності

Publons (рисунок 1.6) орієнтована на облік рецензійної діяльності дослідників і взаємодіє з Web of Science, надаючи розширені можливості оцінювання наукового внеску. Система дозволяє відстежувати рецензування, формувати статистику та підвищувати прозорість наукової комунікації.

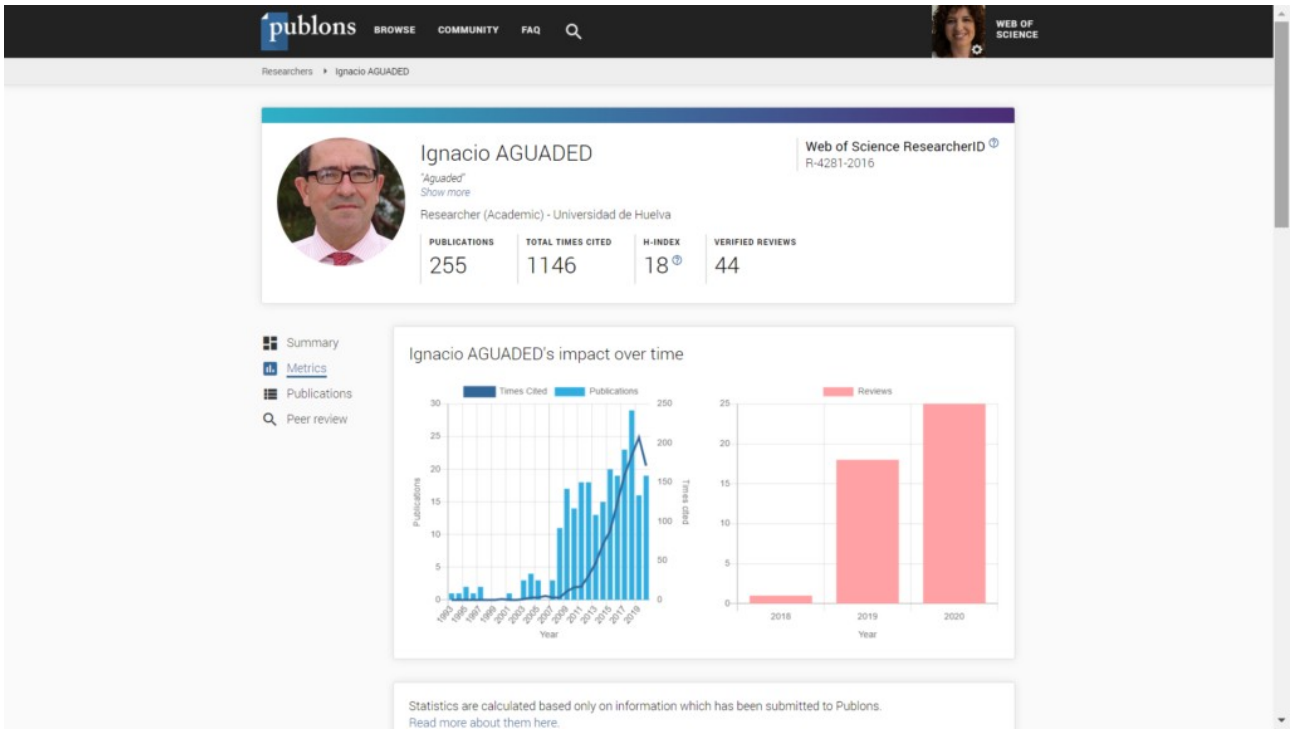


Рисунок 1.6 – Інтерфейс Publons із відображенням статистики наукової діяльності

Для узагальнення характеристик розглянутих систем та визначення їх придатності до використання у сфері обліку діяльності студентських наукових гуртків сформовано порівняльну таблицю (таблиця 1.2), яка відображає основні функціональні можливості та обмеження.

Таблиця 1.2 – Порівняльна характеристика існуючих рішень та розроблюваної системи

Система	Призначення	Основні функції	Недоліки / Обмеження
ORCID	Ідентифікація дослідників	Унікальний профіль, інтеграція даних	Відсутність аналітики діяльності гуртків
Google Scholar	Індексація публікацій	Пошук, метрики цитування	Обмежений контроль даних
Scopus	Аналітика наукових даних	Бібліометричний аналіз, рейтинги	Платний доступ
Web of Science	Оцінювання наукової діяльності	Аналітика, звіти	Складність використання
Publons	Облік рецензування	Статистика рецензій	Вузька спеціалізація

Продовження таблиці 1.2

Розроблювана система	Облік діяльності гуртків	Управління учасниками, аналітика, звіти	Орієнтована на локальне використання

Аналіз існуючих рішень свідчить, що більшість із них орієнтована на облік наукових публікацій та оцінювання індивідуальної дослідницької активності, тоді як аспекти організації діяльності студентських наукових гуртків залишаються поза межами їх функціональних можливостей. Відсутність інструментів для обліку участі студентів, планування заходів, аналізу ефективності гуртків та формування звітності обумовлює необхідність розроблення спеціалізованої інформаційної системи, яка забезпечить комплексну підтримку цих процесів і підвищить ефективність управління науковою діяльністю студентів.

1.3 Моделювання предметної області інформаційної системи обліку та аналізу діяльності студентських наукових гуртків

Моделювання предметної області є ключовим етапом формалізації процесів, що реалізуються в інформаційній системі обліку та аналізу діяльності студентських наукових гуртків. Воно дозволяє визначити функціональні можливості системи, встановити взаємодію користувачів із програмними компонентами, а також описати логіку оброблення даних. Для побудови моделей використано нотацію UML, що забезпечує уніфіковане подання структури системи та динаміки її функціонування.

На рисунку 1.7 наведено діаграму прецедентів, яка відображає основні ролі користувачів і сценарії їхньої взаємодії із системою. Основними акторами є студент, керівник гуртка, адміністратор та представник деканату / Management. Студент може пройти реєстрацію й авторизацію, переглядати наукові гуртки,

подавати заявки на вступ, переглядати власні заявки, відвідуваність, заходи та результати. Керівник гуртка переглядає заявки до своїх гуртків, схвалює або відхиляє їх і веде облік відвідуваності. Адміністратор керує основними даними системи, а представник деканату працює з аналітикою, CSV-експортом і затвердженням звітів.

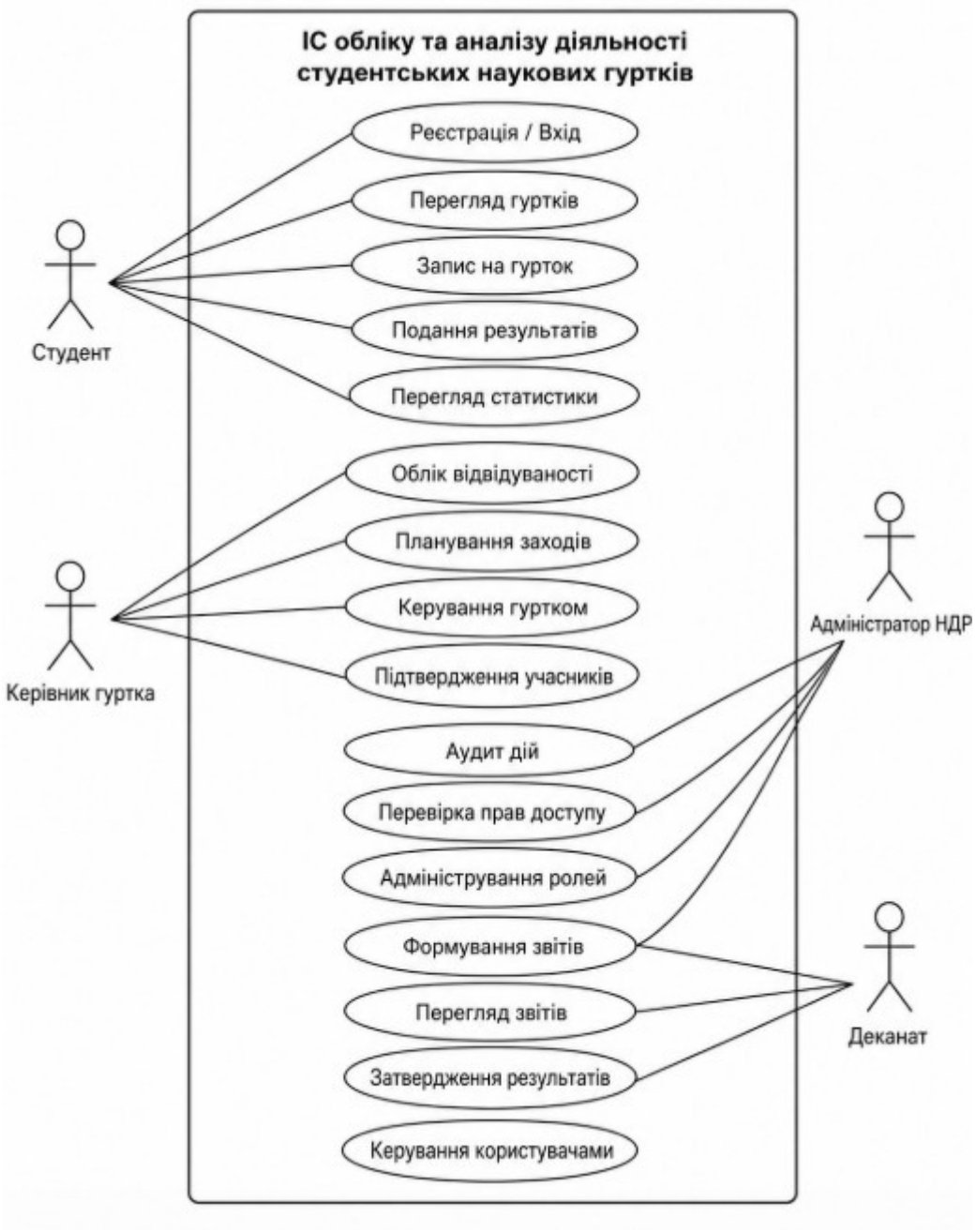


Рисунок 1.7 – Діаграма прецедентів інформаційної системи обліку та аналізу діяльності студентських наукових гуртків

Для деталізації процесу оброблення заявки побудовано діаграму послідовності, наведену на рисунку 1.8. Вона ілюструє подання студентом заявки на вступ до наукового гуртка та її подальший розгляд керівником гуртка або адміністратором. Після подання заявки система перевіряє поточного студента, активність гуртка, наявність чинного членства та дублюючої заявки зі статусом Pending. Якщо перевірки пройдено успішно, заявка зберігається в базі даних зі статусом Pending і стає доступною для розгляду.

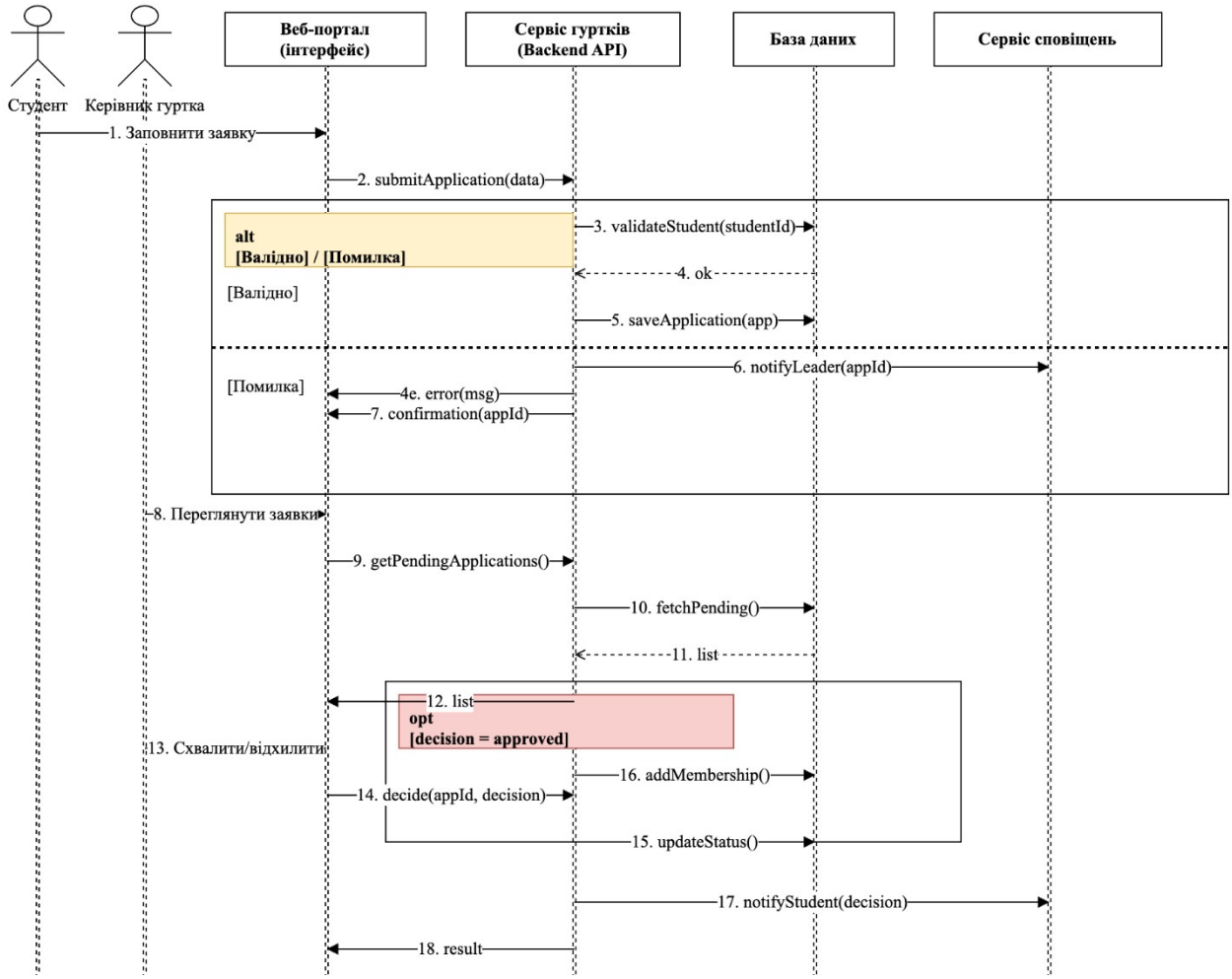


Рисунок 1.8 – Діаграма послідовності оброблення заявки на участь у студентському науковому гуртку

На рисунку 1.9 представлено діаграму діяльності процесу подання та розгляду заявки на участь у студентському науковому гуртку. Процес починається з авторизації студента, вибору наукового гуртка та подання заявки. Далі система перевіряє авторизацію користувача, активність гуртка, наявність чинного

членства та дублюючої заявки зі статусом Pending. У разі успішного проходження перевірок створюється заявка зі статусом Pending. Після цього керівник гуртка або адміністратор розглядає заявку, перевіряє право на розгляд і приймає рішення про схвалення або відхилення. У разі схвалення створюється запис про членство, а статус заявки змінюється на Approved; у разі відхилення заявка отримує статус Rejected.

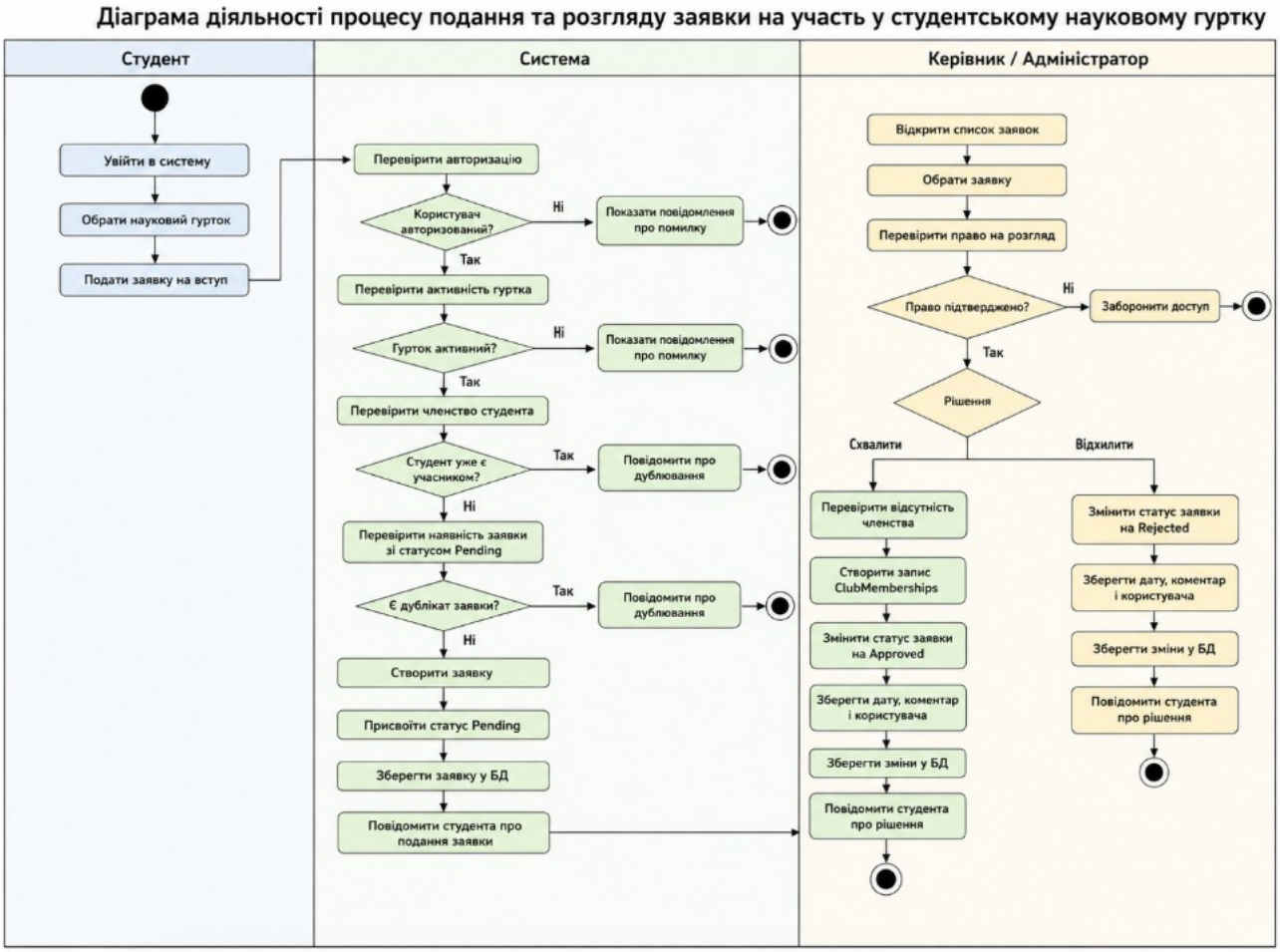


Рисунок 1.9 – Діаграма діяльності процесів управління діяльністю студентських наукових гуртків

Побудовані UML-моделі дозволили структуризувати предметну область та визначити взаємозв’язки між основними компонентами системи. Отримані результати підтверджують доцільність використання модульної архітектури, у якій окремі підсистеми відповідають за автентифікацію користувачів,

управління гуртками, облік діяльності, аналітичну обробку даних і формування звітності.

Результати моделювання є основою для подальшого етапу формування вимог до системи та постановки завдання, у межах яких визначаються функціональні, нефункціональні та технічні характеристики інформаційної системи, а також критерії оцінювання її ефективності.

1.4 Аналіз вимог програмної системи

Аналіз вимог до інформаційної системи обліку та аналізу діяльності студентських наукових гуртків спрямований на формалізацію функціональних, нефункціональних і безпекових характеристик, що визначають архітектуру, продуктивність і надійність програмного продукту. Чітке структурування вимог забезпечує узгоджене проєктування компонентів системи, ефективну взаємодію між ними та відповідність очікуванням користувачів і сучасним принципам програмної інженерії.

Функціональні вимоги визначають основні можливості системи, необхідні для реалізації процесів обліку, управління та аналізу діяльності студентських наукових гуртків. Система повинна забезпечувати реєстрацію та авторизацію користувачів, управління профілями, ведення обліку гуртків і їх учасників, планування заходів, фіксацію результатів діяльності, а також формування аналітичних звітів. Узагальнені функціональні вимоги наведено в таблиці 1.3.

Таблиця 1.3 – Функціональні вимоги до інформаційної системи обліку та аналізу діяльності студентських наукових гуртків

№	Вимога	Опис	Очікуваний результат
1	Реєстрація та авторизація	Надання доступу до системи через обліковий запис	Безпечний доступ користувачів
2	Управління профілем	Редагування персональних даних користувача	Актуальність інформації
3	Управління гуртками	Створення, редагування та перегляд інформації про гуртки	Централізований облік гуртків

4	Облік учасників	Додавання, підтвердження та керування учасниками	Актуальний склад гуртків
5	Планування заходів	Формування календаря подій, опис заходів	Організація діяльності гуртків
6	Облік результатів	Фіксація публікацій, проєктів, досягнень	Накопичення наукових результатів

Нефункціональні вимоги визначають якісні характеристики функціонування системи, зокрема її продуктивність, надійність, масштабованість, зручність використання та сумісність. Вони забезпечують стабільну роботу системи в умовах багатокористувацького доступу та відповідність сучасним технологічним стандартам. Основні нефункціональні вимоги наведено в таблиці 1.4.

Таблиця 1.4 – Нефункціональні вимоги до інформаційної системи обліку та аналізу діяльності студентських наукових гуртків

№	Категорія	Вимога	Показник / Критерій
1	Продуктивність	Обробка користувацьких запитів	≤ 2 секунд
2	Масштабованість	Підтримка одночасної роботи користувачів	не менше 100 користувачів
3	Надійність	Безперервність роботи системи	≥ 99,5 % доступності
4	Зручність інтерфейсу	Інтуїтивно зрозумілий інтерфейс	Мінімальна кількість дій
5	Сумісність	Робота в сучасних браузерах	Chrome, Firefox, Edge
6	Розширюваність	Можливість додавання нових модулів	Модульна архітектура

Вимоги до безпеки визначають механізми захисту інформації, персональних даних користувачів та результатів наукової діяльності. Особлива увага приділяється автентифікації, контролю доступу, захисту каналів передачі даних та резервному копіюванню. Основні вимоги до безпеки системи наведено в таблиці 1.5.

Таблиця 1.5 – Вимоги до безпеки інформаційної системи обліку та аналізу діяльності студентських наукових гуртків

№	Вимога	Механізм реалізації	Очікуваний результат
1	Автентифікація та авторизація	Рольова модель доступу	Захищений доступ до системи
2	Захист каналів зв’язку	Використання HTTPS	Конфіденційність даних
3	Захист персональних даних	Шифрування паролів (bcrypt)	Безпечне зберігання даних

Проведений аналіз вимог дозволяє визначити ключові пріоритети розроблення системи, серед яких забезпечення зручності використання, надійності функціонування, захисту даних та ефективної аналітичної підтримки. Сформульовані вимоги є основою для наступного етапу – постановки завдання, у межах якого деталізуються структура системи, алгоритми оброблення даних і принципи реалізації її функціональних компонентів.

1.5 Постановка завдання

Постановка завдання визначає мету, вхідні та вихідні дані, а також загальні принципи функціонування інформаційної системи обліку та аналізу діяльності студентських наукових гуртків. На основі проведеного аналізу предметної області, результатів моделювання та сформульованих вимог визначено основні задачі, які система повинна реалізувати для забезпечення ефективного обліку, управління та аналітичної підтримки наукової діяльності студентів.

Метою розробки є створення інформаційної системи, що забезпечує автоматизований облік діяльності студентських наукових гуртків, управління учасниками та заходами, а також формування аналітичних звітів і показників ефективності наукової діяльності.

Вхідні дані:

– персональні дані користувачів (студентів, керівників гуртків, адміністраторів), необхідні для реєстрації та авторизації в системі;

– інформація про студентські наукові гуртки (назва, тематика, опис, склад учасників);

– дані про заходи (плани, календар подій, програми, протоколи);

– результати наукової діяльності студентів (публікації, проєкти, доповіді, участь у конференціях);

– дані з університетських інформаційних ресурсів (реєстри студентів, академічні показники, групи);

– журнали подій та історія взаємодії користувачів із системою.

Вихідні дані:

– актуальна інформація про склад гуртків і участь студентів у науковій діяльності;

– аналітичні показники ефективності діяльності гуртків (активність учасників, кількість заходів, результати);

– зведені та деталізовані звіти за гуртками, кафедрами або визначеними періодами;

– повідомлення користувачам через інтерфейс системи щодо результатів виконаних дій, подання заявок та їх розгляду

– експортовані звіти у форматі CSV із можливістю подальшого розширення до форматів PDF або Excel.

Основні задачі системи:

1. розроблення користувацького інтерфейсу з підтримкою реєстрації, авторизації та керування профілем користувача.
2. реалізація функціоналу управління студентськими науковими гуртками, включаючи створення, редагування та перегляд інформації.
3. забезпечення обліку учасників гуртків, оброблення заявок на вступ та керування ролями користувачів.
4. розроблення модуля планування та проведення заходів із можливістю фіксації результатів діяльності.
5. реалізація підсистеми аналітики для оцінювання ефективності діяльності гуртків і активності студентів.
6. забезпечення формування звітів за різними критеріями (гурток, кафедра, період) з можливістю експорту.

7. розроблення механізмів сповіщення користувачів про події, рішення та зміни в системі.

8. реалізація адміністративного модуля для керування користувачами, ролями, довідниками та контролю доступу.

9. забезпечення цілісності, надійності та захисту даних під час їх оброблення, передачі та зберігання.

Очікувані результати полягають у створенні інформаційної системи, яка забезпечує швидкий доступ до актуальної інформації, підвищує ефективність обліку діяльності студентських наукових гуртків, автоматизує процеси формування звітності та сприяє прийняттю обґрунтованих управлінських рішень. Система повинна підтримувати багатокористувацький режим роботи, забезпечувати високу продуктивність і зручність використання, а також інтеграцію з іншими інформаційними ресурсами університету.

Сформульоване завдання охоплює повний цикл функціонування інформаційної системи – від збору та оброблення вхідних даних до формування аналітичних результатів і звітності. Реалізація поставлених вимог дозволить створити комплексний інструмент для підтримки організації та аналізу наукової діяльності студентів, що сприятиме підвищенню якості освітньо-наукового процесу у закладах вищої освіти.

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Логічна модель даних

Логічна модель даних інформаційної системи визначає структуру збереження інформації, яка використовується в програмній системі. Вона описує основні таблиці бази даних, їхні атрибути, первинні та зовнішні ключі, а також зв’язки між сутностями предметної області.

Модель побудована за реляційним принципом. Дані зберігаються у взаємопов’язаних таблицях, кожна з яких має власне функціональне призначення. Первинні ключі забезпечують унікальну ідентифікацію записів, а зовнішні ключі підтримують цілісність і узгодженість даних.

Структура бази даних реалізована у програмному проєкті за допомогою контексту StudentScienceClubsContext та міграції InitialSchema, яка створює таблиці, ключі, індекси й обмеження унікальності.

Організаційна частина моделі представлена таблицями Faculties, Departments, StudyGroups, Students та Supervisors. Вони зберігають дані про факультети, кафедри, академічні групи, студентів і керівників гуртків, що дозволяє відобразити структуру закладу освіти.

Центральною сутністю є таблиця ScienceClubs, яка містить інформацію про студентські наукові гуртки, їхню тематику, опис, статус активності, кафедру та керівника. Для обліку участі студентів використовується таблиця ClubMemberships, а статуси членства зберігаються в довіднику MembershipStatuses.

Заявки на вступ до гуртків фіксуються в таблиці ClubApplications. Вона містить дані про студента, гурток, дату подання, статус розгляду, коментарі, дату розгляду та користувача, який опрацював заявку.

Поточна діяльність гуртків обліковується за допомогою таблиць ClubMeetings, Attendances, ScientificEvents, EventParticipants та

ParticipationResults. Вони забезпечують збереження даних про засідання, відвідуваність, наукові заходи, учасників і результати участі студентів.

Окремий блок становлять таблиці AppUsers, AppLoginLogs та ReportApprovals. AppUsers зберігає облікові записи користувачів, AppLoginLogs використовується для аудиту входів, а ReportApprovals — для збереження затверджених аналітичних звітів.

Крім таблиць, у базі даних використовується представлення vw_ClubActivitySummary, яке узагальнює інформацію про активність гуртків: кількість активних учасників, наукових заходів і результатів діяльності. Це спрощує формування аналітичної інформації для звітності.

Основні таблиці логічної моделі даних наведено в таблиці 2.1.

Таблиця 2.1 – Основні таблиці логічної моделі даних

Таблиця	**Призначення**
Faculties	Зберігання інформації про факультети
Departments	Зберігання інформації про кафедри
StudyGroups	Облік академічних груп
Students	Зберігання даних про студентів
Supervisors	Зберігання даних про керівників наукових гуртків
ScienceClubs	Облік студентських наукових гуртків
MembershipStatuses	Довідник статусів членства
ClubMemberships	Облік участі студентів у гуртках
ClubMeetings	Облік засідань наукових гуртків
EventTypes	Довідник типів наукових заходів
EventStatuses	Довідник статусів наукових заходів
ScientificEvents	Облік наукових заходів
ParticipationRoles	Довідник ролей участі у заходах
EventParticipants	Облік учасників наукових заходів
ParticipationResults	Зберігання результатів участі студентів
AppUsers	Облік користувачів системи

Таблиця	Призначення
AppLoginLogs	Журнал входів користувачів
ClubApplications	Облік заявок на вступ до гуртків
Attendances	Облік відвідуваності засідань
ReportApprovals	Збереження затверджених аналітичних звітів

Основні зв’язки між таблицями логічної моделі наведено в таблиці 2.2.

Таблиця 2.2 – Основні зв’язки логічної моделі даних

Зв’язок	Тип зв’язку	Опис
Faculties — Departments	1:M	Один факультет може містити декілька кафедр
Departments — StudyGroups	1:M	Одна кафедра може мати декілька академічних груп
Departments — Supervisors	1:M	Одна кафедра може мати декілька керівників гуртків
Departments — ScienceClubs	1:M	Одна кафедра може мати декілька наукових гуртків
StudyGroups — Students	1:M	Одна академічна група може містити декілька студентів
Supervisors — ScienceClubs	1:M	Один керівник може керувати декількома гуртками
ScienceClubs — ClubMemberships	1:M	Один гурток може мати багато записів про членство
Students — ClubMemberships	1:M	Один студент може бути учасником декількох гуртків
MembershipStatuses — ClubMemberships	1:M	Один статус може використовуватися в багатьох записах членства
ScienceClubs — ClubMeetings	1:M	Один гурток може мати багато засідань
ClubMeetings — Attendances	1:M	Одне засідання може мати багато записів відвідуваності
ScienceClubs — ScientificEvents	1:M	Один гурток може організовувати багато наукових заходів
EventTypes — ScientificEvents	1:M	Один тип може застосовуватися до багатьох заходів

Зв’язок	Тип зв’язку	Опис
EventStatuses — ScientificEvents	1:M	Один статус може застосовуватися до багатьох заходів
ScientificEvents — EventParticipants	1:M	Один захід може мати багато учасників
Students — EventParticipants	1:M	Один студент може брати участь у багатьох заходах
ParticipationRoles — EventParticipants	1:M	Одна роль може використовуватися в багатьох записах участі
EventParticipants — ParticipationResults	1:M	Один учасник заходу може мати декілька результатів
ScienceClubs — ClubApplications	1:M	Один гурток може отримувати багато заявок
Students — ClubApplications	1:M	Один студент може подавати декілька заявок
AppUsers — ClubApplications	1:M	Один користувач може розглянути декілька заявок
AppUsers — AppLoginLogs	1:M	Один користувач може мати багато записів входу
AppUsers — ReportApprovals	1:M	Один користувач може затвердити декілька звітів
Faculties — ReportApprovals	1:M	Один факультет може бути пов’язаний із декількома звітами
ScienceClubs — ReportApprovals	1:M	Один гурток може бути пов’язаний із декількома звітами

ER-діаграму логічної моделі даних інформаційної системи наведено на рисунку 2.1.

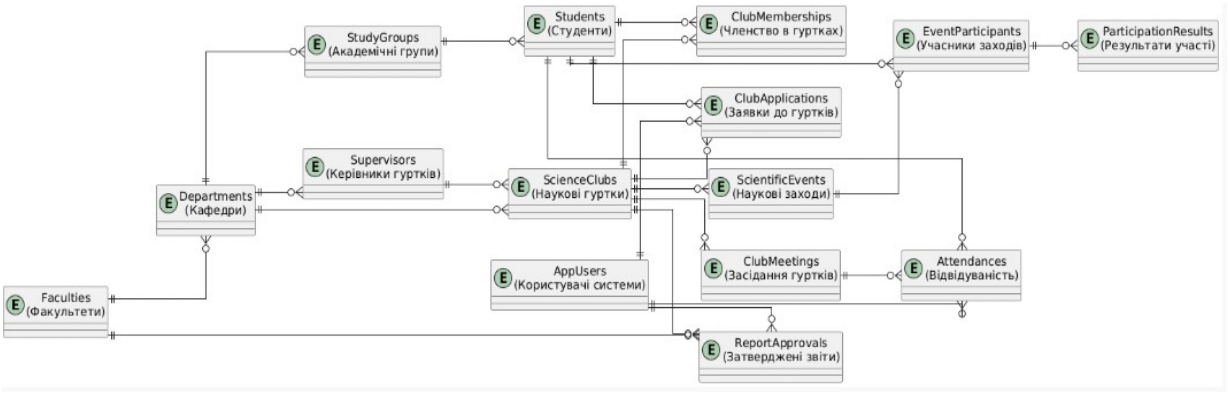


Рисунок 2.1 – Спрощена ER-діаграма логічної моделі даних інформаційної системи обліку та аналізу діяльності студентських наукових гуртків

Побудована логічна модель відповідає вимогам реляційної структури даних. Основні таблиці мають первинні ключі, а зв’язки між ними реалізуються за допомогою зовнішніх ключів. Дані розподілені за окремими таблицями відповідно до їхнього функціонального призначення, що дозволяє уникнути надлишкового дублювання інформації.

Для забезпечення унікальності та коректності даних у базі передбачено додаткові індекси й унікальні обмеження. Зокрема, унікальними є логін користувача, електронна пошта студента, номер залікової книжки, код академічної групи, назви довідникових значень, назва наукового гуртка, а також окремі комбінації полів, що запобігають дублюванню записів про членство, участь у заходах та відвідуваність.

Запропонована модель забезпечує підтримку основних процесів системи: облік студентів, ведення інформації про наукові гуртки, опрацювання заявок, фіксацію членства, планування засідань і заходів, облік відвідуваності, збереження результатів діяльності та формування аналітичних звітів.

Таким чином, логічна модель даних є основою для створення фізичної структури бази даних, вибору системи управління базами даних та подальшої реалізації програмних модулів інформаційної системи.

2.2 Вибір системи управління інформаційною базою

Після побудови логічної моделі даних необхідно обрати систему управління базою даних, яка забезпечить надійне зберігання, оброблення та захист інформації. Вибір СУБД є важливим етапом проєктування, оскільки саме від нього залежить стабільність роботи інформаційної системи, швидкість доступу до даних, підтримка зв’язків між таблицями та можливість подальшого розширення програмного продукту.

Інформаційна система обліку та аналізу діяльності студентських наукових гуртків працює з великою кількістю взаємопов’язаних сутностей. До них належать факультети, кафедри, академічні групи, студенти, керівники гуртків, наукові гуртки, заявки на участь, засідання, відвідуваність, наукові заходи, результати діяльності, користувачі системи та аналітичні звіти. Тому для реалізації інформаційної бази доцільно використовувати реляційну СУБД, яка підтримує первинні та зовнішні ключі, індекси, унікальні обмеження і представлення.

У розробленому програмному проєкті для роботи з базою даних використовується Microsoft SQL Server Express. Це підтверджується рядком підключення у файлі appsettings.json, де вказано локальний екземпляр SQLEXPRESS та базу даних StudentScienceClubsDB. Також у файлі Program.cs підключення до бази даних виконується через метод UseSqlServer, що вказує на використання провайдера Microsoft.EntityFrameworkCore.SqlServer.

Основними критеріями вибору системи управління базою даних для даного проєкту є: підтримка реляційної моделі даних; підтримка первинних і зовнішніх ключів; можливість створення індексів та унікальних обмежень; підтримка представлень для аналітичної обробки даних; інтеграція з ASP.NET Core; підтримка Entity Framework Core; можливість використання міграцій для створення та оновлення структури бази даних; зручність адміністрування; достатня продуктивність для навчального та прикладного проєкту.

Для порівняння можливих варіантів СУБД розглянуто Microsoft SQL Server Express, SQLite, MySQL та PostgreSQL. Порівняльну характеристику наведено в таблиці 2.3.

Таблиця 2.3 – Порівняння систем управління базами даних

СУБД	**Переваги**	**Обмеження / особливості**	**Доцільність для проєкту**
Microsoft SQL Server Express	Добра інтеграція з ASP.NET Core та Entity Framework Core; підтримка	Має обмеження редакції Express, однак їх достатньо для дипломного	Найбільш доцільна для даної системи.

	зовнішніх ключів, індексів, унікальних обмежень, представлень і міграцій.	проєкту.	
SQLite	Простота використання; не потребує окремого сервера; легко підключається до невеликих застосунків.	Менш придатна для багатокористувацьких інформаційних систем і складної аналітики.	Більше підходить для малих локальних застосунків.
MySQL	Поширена СУБД для вебпроєктів; підтримує реляційні дані та індекси.	У даному проєкті не використовується; потребує іншого провайдера Entity Framework Core.	Можлива альтернатива, але не є основним вибором.
PostgreSQL	Потужна, надійна та розширювана СУБД; підтримує складні запити.	Потребує іншого провайдера та додаткового налаштування проєкту.	Сильна альтернатива, але в цьому проєкті не використовується.

За результатами порівняння для реалізації інформаційної бази було обрано Microsoft SQL Server Express. Ця СУБД є доцільною для даного проєкту, оскільки система має чітко структуровану реляційну модель даних і вимагає збереження коректних зв’язків між таблицями. Наприклад, студенти пов’язані з академічними групами, гуртки — з кафедрами та керівниками, заявки — зі студентами й гуртками, а результати діяльності — з участю студентів у наукових заходах.

Важливою перевагою Microsoft SQL Server Express є підтримка механізмів забезпечення цілісності даних. У базі даних використовуються первинні ключі для унікальної ідентифікації записів, зовнішні ключі для встановлення зв’язків між таблицями, а також індекси та унікальні обмеження для запобігання дублюванню важливих даних. Наприклад, у проєкті передбачено унікальність логіна користувача, електронної пошти студента, номера залікової книжки, коду академічної групи та назви наукового гуртка.

Ще однією перевагою є інтеграція Microsoft SQL Server Express з платформою ASP.NET Core та Entity Framework Core. У проєкті використовується контекст бази даних StudentScienceClubsContext, який містить набори сутностей для роботи з таблицями бази даних. Через Entity Framework Core забезпечується типобезпечна робота з даними, використання LINQ-запитів,

а також застосування міграцій для створення та оновлення структури бази даних.

У міграції InitialSchema визначено структуру таблиць, типи даних, первинні ключі, зовнішні ключі, індекси та унікальні обмеження. Використання типів nvarchar, datetime2, bit, decimal, а також механізму SqlServer:Identity підтверджує орієнтацію структури бази даних саме на Microsoft SQL Server.

Для аналітичної частини системи важливою є підтримка представлень. У проєкті використовується представлення vw_ClubActivitySummary, яке узагальнює інформацію про активність наукових гуртків. Воно дозволяє отримувати зведені дані про кількість учасників, заходів та результатів діяльності без необхідності кожного разу формувати складні запити вручну.

Таким чином, Microsoft SQL Server Express є оптимальним вибором для інформаційної системи обліку та аналізу діяльності студентських наукових гуртків. Вона забезпечує надійне зберігання реляційних даних, підтримує механізми цілісності, індексації, унікальності, представлення, а також добре інтегрується з ASP.NET Core та Entity Framework Core. Обрана СУБД повністю відповідає потребам розробленої системи та створює основу для подальшого розвитку програмного продукту.

2.3 Створення інформаційної бази

Створення інформаційної бази є наступним етапом після побудови логічної моделі даних та вибору системи управління базою даних. На цьому етапі логічна структура даних перетворюється на фізичну структуру бази даних, яка використовується програмною системою для зберігання, оброблення та отримання інформації.

У розробленій інформаційній системі база даних створюється з використанням Microsoft SQL Server Express та технології Entity Framework Core. У файлі appsettings.json задано рядок підключення до локального

екземпляра SQLEXPRESS і бази даних StudentScienceClubsDB. Це дозволяє програмному застосунку встановлювати з’єднання з інформаційною базою під час запуску системи.

Інформаційна база системи має централізований характер, оскільки всі основні дані зберігаються в єдиній базі даних StudentScienceClubsDB на сервері Microsoft SQL Server Express. Користувачі не працюють із базою даних безпосередньо, а отримують доступ до інформації через програмний застосунок. Такий підхід дозволяє централізовано контролювати збереження даних, доступ користувачів, оновлення структури бази та формування резервних копій.

Підключення контексту бази даних до програмного застосунку виконується у файлі Program.cs. Для цього використовується метод AddDbContext, у якому контекст StudentScienceClubsContext підключається до Microsoft SQL Server за допомогою методу UseSqlServer. Це забезпечує взаємодію між програмною частиною системи та базою даних через Entity Framework Core.

Контекст бази даних StudentScienceClubsContext містить набори сутностей DbSet, які відповідають основним таблицям інформаційної бази. До них належать таблиці студентів, факультетів, кафедр, академічних груп, керівників гуртків, наукових гуртків, членства, заявок, засідань, відвідуваності, наукових заходів, учасників заходів, результатів участі, користувачів системи та затверджених звітів. Крім того, у контексті визначено зв’язки між сутностями, первинні ключі, зовнішні ключі та окремі властивості таблиць.

Фізична структура бази даних створюється за допомогою міграції InitialSchema. У цій міграції визначено створення основних таблиць, їхніх полів, типів даних, первинних ключів, зовнішніх ключів, індексів та унікальних обмежень. Використання міграцій дозволяє автоматизувати процес створення й

оновлення бази даних, а також забезпечує відповідність структури бази даних моделям програмного застосунку.

У процесі створення інформаційної бази використовуються типи даних, характерні для Microsoft SQL Server, зокрема int, nvarchar, date, datetime2, bit, decimal, tinyint та smallint. Для автоматичного формування ідентифікаторів записів використовується механізм SqlServer:Identity. Це дозволяє автоматично створювати унікальні значення первинних ключів для записів у таблицях.

Основні етапи створення інформаційної бази наведено в таблиці 2.4.

Таблиця 2.4 – Етапи створення інформаційної бази

Етап	Опис
Налаштування підключення	У файлі appsettings.json задається рядок підключення до бази даних StudentScienceClubsDB
Підключення контексту	У файлі Program.cs реєструється StudentScienceClubsContext через метод UseSqlServer
Опис сутностей	У StudentScienceClubsContext визначаються набори сутностей, що відповідають таблицям бази даних
Створення міграції	Міграція InitialSchema описує структуру таблиць, ключів, індексів і зв’язків
Застосування міграції	Під час запуску системи виконується оновлення структури бази даних через MigrateAsync
Створення службових об’єктів	У базі створюються індекси, унікальні обмеження та представлення для аналітики
Початкове заповнення	У системі створюються базові облікові записи користувачів для роботи з різними ролями

У базі даних створюються таблиці, які відповідають основним сутностям предметної області. Організаційна структура представлена таблицями Faculties, Departments, StudyGroups, Students та Supervisors. Вони забезпечують зберігання даних про факультети, кафедри, академічні групи, студентів і керівників наукових гуртків.

Для обліку студентських наукових гуртків використовується таблиця ScienceClubs, яка пов’язана з кафедрою та керівником гуртка. Участь студентів у гуртках зберігається в таблиці ClubMemberships, а статуси членства — у таблиці MembershipStatuses. Для обліку заявок на вступ до гуртків створюється таблиця ClubApplications.

Облік діяльності гуртків реалізовано за допомогою таблиць ClubMeetings, Attendances, ScientificEvents, EventParticipants та ParticipationResults. Таблиця ClubMeetings зберігає інформацію про засідання гуртків, а таблиця Attendances використовується для фіксації відвідуваності студентів. Таблиця ScientificEvents призначена для зберігання наукових заходів, EventParticipants — для обліку учасників заходів, а ParticipationResults — для фіксації результатів участі студентів.

Окремий блок інформаційної бази становлять таблиці AppUsers, AppLoginLogs та ReportApprovals. Таблиця AppUsers використовується для зберігання облікових записів користувачів системи. Таблиця AppLoginLogs фіксує інформацію про входи користувачів у систему. Таблиця ReportApprovals призначена для збереження затверджених аналітичних звітів.

У процесі створення бази даних особлива увага приділяється забезпеченню цілісності даних. Для цього в таблицях використовуються первинні ключі, які забезпечують унікальну ідентифікацію записів, та зовнішні ключі, які встановлюють зв’язки між таблицями. Наприклад, студенти пов’язані з академічними групами, наукові гуртки — з кафедрами та керівниками, заявки — зі студентами й гуртками, а результати участі — з учасниками наукових заходів.

Крім ключів, у базі даних використовуються індекси та унікальні обмеження. Вони дозволяють підвищити швидкість пошуку даних і запобігти дублюванню важливої інформації. Зокрема, унікальними є логін користувача, електронна пошта студента, номер залікової книжки, код академічної групи,

назва факультету, назва наукового гуртка та окремі комбінації полів, які відповідають за унікальність членства, відвідуваності й участі у заходах.

Основні об’єкти, що створюються в інформаційній базі, наведено в таблиці 2.5.

Таблиця 2.5 – Основні об’єкти інформаційної бази

Об’єкт інформаційної бази	Призначення
Таблиці	Зберігання основних даних системи
Первинні ключі	Унікальна ідентифікація записів
Зовнішні ключі	Встановлення зв’язків між таблицями
Індекси	Прискорення пошуку та обробки даних
Унікальні обмеження	Запобігання дублюванню важливих записів
Представлення	Формування узагальненої аналітичної інформації
Міграції	Автоматизоване створення та оновлення структури бази даних

Для аналітичної частини системи використовується представлення vw_ClubActivitySummary. Воно узагальнює дані про активність студентських наукових гуртків, зокрема кількість активних учасників, кількість наукових заходів і кількість результатів діяльності. Використання представлення дозволяє спростити формування звітів та зменшити складність запитів до бази даних.

Представлення vw_ClubActivitySummary створюється окремою міграцією AddClubActivitySummaryView за допомогою SQL-команди CREATE OR ALTER VIEW. Воно об’єднує дані з таблиць наукових гуртків, кафедр, керівників, членства, наукових заходів та результатів участі. Завдяки цьому система може швидко отримувати зведену інформацію для аналітичних звітів.

У файлі Program.cs також реалізовано механізм автоматичного застосування міграцій під час запуску системи. Для цього використовується метод context.Database.MigrateAsync(), який забезпечує приведення структури бази даних у відповідність до поточних міграцій проєкту. Такий підхід спрощує

розгортання системи та зменшує ризик невідповідності між програмними моделями й реальною структурою бази даних.

Також у програмному коді передбачено початкове створення облікових записів користувачів. Зокрема, створюються користувачі з ролями адміністратора, студента, керівника гуртка та представника деканату. Це дозволяє після запуску системи одразу перевірити роботу механізмів авторизації, розмежування доступу та взаємодії з основними модулями системи.

На рисунку 2.3 наведено фрагмент файлу Program.cs, який демонструє підключення контексту бази даних до Microsoft SQL Server.

```
builder.Services.AddDbContext<StudentScienceClubsContext>(options =>
    options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnect
```

Рисунок 2.3 – Підключення контексту бази даних до Microsoft SQL Server у файлі Program.cs

На рисунку 2.4 наведено фрагмент файлу Program.cs, у якому виконується автоматичне застосування міграцій бази даних під час запуску системи.

```
    await MarkExistingDatabaseAsMigratedAsync(context);
    await context.Database.MigrateAsync();
}
```

Рисунок 2.4 – Застосування міграцій бази даних у файлі Program.cs

Наведені фрагменти коду підтверджують, що інформаційна база підключається до програмної системи через Microsoft SQL Server, а її структура оновлюється засобами Entity Framework Core. Метод UseSqlServer забезпечує підключення до бази даних, а метод MigrateAsync застосовує наявні міграції та приводить структуру бази даних до актуального стану.

Таким чином, інформаційна база системи створюється на основі реляційної моделі даних із використанням Microsoft SQL Server Express та Entity Framework Core. Застосування міграцій забезпечує автоматизоване формування таблиць, зв’язків, ключів, індексів, обмежень та аналітичного представлення. Створена інформаційна база забезпечує зберігання даних про студентів, наукові

гуртки, заявки, засідання, відвідуваність, наукові заходи, результати діяльності та звітність, що є основою для функціонування всієї інформаційної системи.

3 ПРИКЛАДНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Організаційна структура програмного забезпечення

Прикладне програмне забезпечення інформаційної системи обліку та аналізу діяльності студентських наукових гуртків реалізовано у вигляді вебзастосунку на платформі ASP.NET Core з використанням архітектурного підходу MVC. Такий підхід передбачає поділ програмного забезпечення на моделі даних, контролери та представлення, що дозволяє логічно розмежувати структуру даних, обробку запитів користувача та формування інтерфейсу.

Організаційна структура програмного забезпечення побудована за монолітним принципом. У межах одного вебзастосунку об’єднано користувацький інтерфейс, серверну логіку, роботу з базою даних, механізми авторизації, модулі обліку та аналітичні функції. Такий підхід є доцільним для бакалаврської роботи, оскільки система має чітко визначену предметну область і не потребує поділу на окремі мікросервіси.

Користувацький інтерфейс системи реалізовано за допомогою Razor Views. Через вебсторінки користувачі можуть переглядати інформацію, подавати заявки, працювати з гуртками, засіданнями, заходами, результатами участі та аналітичними звітами. Статичні ресурси інтерфейсу, зокрема стилі, скрипти та допоміжні файли, розміщуються в папці wwwroot.

Серверна частина програмного забезпечення представлена контролерами, які розміщено в папці Controllers. Контролери приймають запити користувачів, виконують необхідну перевірку даних, звертаються до бази даних через контекст Entity Framework Core та повертають результат у відповідне представлення. Основна прикладна логіка системи зосереджена саме в контролерах.

Моделі даних і допоміжні моделі представлення розміщено в папці Models. Вони описують основні сутності предметної області: студентів, керівників гуртків, наукові гуртки, заявки, членство, засідання, відвідуваність, наукові заходи, учасників заходів, результати участі, користувачів системи та

затверджені звіти. Також у цій частині проєкту використовуються ViewModel-класи, які призначені для передавання даних між контролерами та представленнями.

Робота з базою даних реалізована через контекст StudentScienceClubsContext, який розміщується в папці Data. Контекст забезпечує взаємодію програмної системи з базою даних Microsoft SQL Server за допомогою Entity Framework Core. Структура бази даних підтримується міграціями, які зберігаються в папці Migrations.

Налаштування запуску вебзастосунку, підключення бази даних, маршрутизація, механізми аутентифікації та авторизації визначаються у файлі Program.cs. Саме в цьому файлі задаються основні параметри роботи системи, підключаються необхідні служби та визначається маршрут обробки запитів.

Функціонально програмне забезпечення поділяється на такі основні модулі: модуль авторизації користувачів, модуль обліку студентів, модуль обліку наукових гуртків, модуль заявок, модуль членства в гуртках, модуль засідань, модуль відвідуваності, модуль наукових заходів, модуль учасників заходів, модуль результатів участі, модуль користувачів системи, модуль журналу входів, а також модуль аналітики та звітності.

Основні структурні елементи програмного забезпечення наведено в таблиці 3.1.

Таблиця 3.1 – Основні структурні елементи програмного забезпечення

Елемент структури	**Призначення**
Program.cs	Налаштування запуску застосунку, маршрутизації, авторизації та підключення бази даних
Controllers	Обробка запитів користувачів і реалізація прикладної логіки
Models	Опис сутностей бази даних і ViewModel-класів
Views	Формування вебінтерфейсу користувача
Data	Зберігання контексту бази даних StudentScienceClubsContext
Migrations	Зберігання міграцій для створення й оновлення структури бази даних
wwwroot	Зберігання статичних файлів інтерфейсу

appsettings.json	Зберігання конфігураційних параметрів застосунку

Взаємодія компонентів системи відбувається за типовою схемою MVC. Користувач відкриває сторінку у браузері, після чого запит передається до серверної частини ASP.NET Core. Маршрутизація визначає відповідний контролер, який обробляє запит, за потреби звертається до бази даних через StudentScienceClubsContext, отримує або змінює дані та передає результат у Razor-представлення. Після цього користувач отримує сформовану HTML-сторінку.

Організаційну структуру програмного забезпечення інформаційної системи наведено на рисунку 3.1.

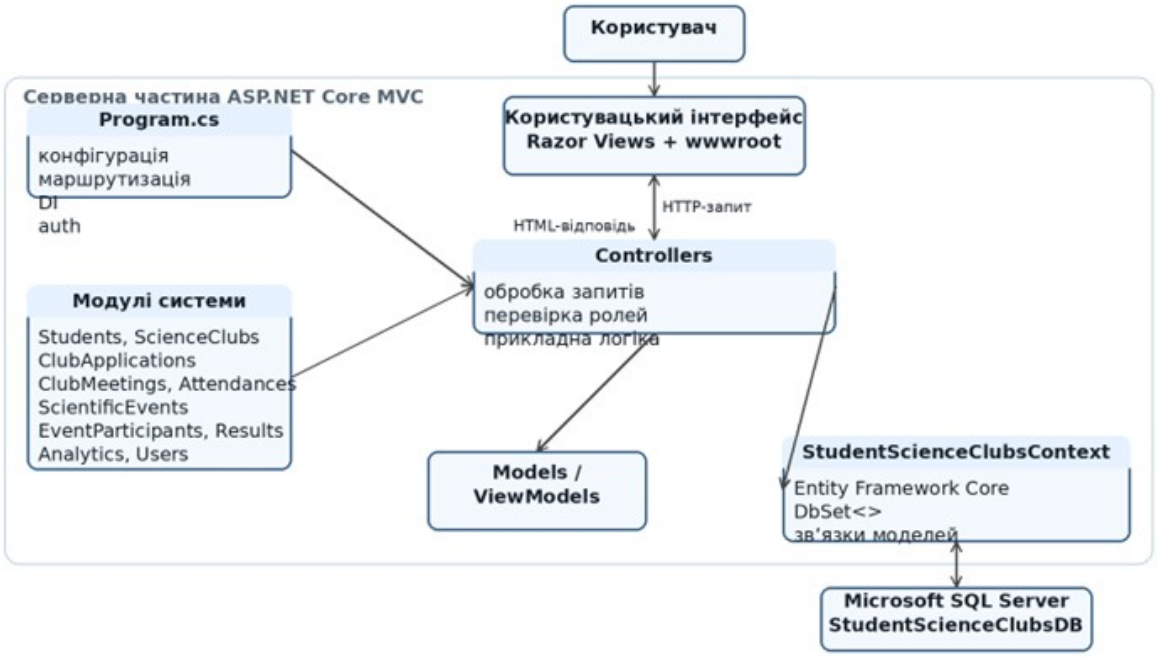


Рисунок 3.1 – Організаційна структура програмного забезпечення інформаційної системи

Наведена структура показує, що програмне забезпечення має чіткий поділ на інтерфейс користувача, серверну логіку, модулі системи, моделі даних, контекст бази даних і саму базу даних. Це забезпечує зрозумілу організацію проєкту, спрощує супровід програмного коду та дозволяє надалі розширювати функціональні можливості системи.

3.2 Вибір інструментарію для створення ППЗ

Для створення прикладного програмного забезпечення інформаційної системи обліку та аналізу діяльності студентських наукових гуртків було обрано інструментарій платформи .NET. Основною мовою програмування є C#, яка використовується для реалізації серверної логіки, моделей даних, контролерів, контексту бази даних, міграцій та налаштування роботи програмного застосунку.

Програмну систему реалізовано як вебзастосунок на основі ASP.NET Core MVC. Такий підхід дозволяє розділити програмне забезпечення на три основні частини: моделі, контролери та представлення. Моделі описують структуру даних, контролери обробляють запити користувачів і виконують прикладну логіку, а представлення відповідають за формування вебсторінок інтерфейсу користувача.

Використання ASP.NET Core MVC є доцільним для розроблюваної системи, оскільки вона має багатосторінковий інтерфейс, різні ролі користувачів і значну кількість функціональних модулів. Система повинна забезпечувати авторизацію, облік студентів, наукових гуртків, заявок, засідань, відвідуваності, наукових заходів, результатів участі та формування аналітичних звітів. Архітектура MVC дозволяє логічно розподілити ці модулі та спростити подальший супровід програмного коду.

Для побудови інтерфейсу користувача використано Razor Views, HTML, CSS і JavaScript. Razor Views забезпечують серверне формування HTML-сторінок на основі даних, отриманих від контролерів. Для оформлення сторінок і створення адаптивного інтерфейсу застосовується Bootstrap. Також у проєкті використовуються jQuery та jQuery Validation, які забезпечують допоміжну клієнтську взаємодію та перевірку правильності введення даних у формах.

Робота з базою даних реалізована за допомогою Entity Framework Core. Цей інструмент дозволяє працювати з таблицями бази даних через класи C# і контекст бази даних StudentScienceClubsContext. Використання Entity

Framework Core зменшує потребу в ручному написанні SQL-запитів, спрощує роботу зі зв’язками між сутностями та забезпечує підтримку міграцій для створення й оновлення структури бази даних.

Як систему управління базою даних використано Microsoft SQL Server Express. Вона підходить для розроблюваної інформаційної системи, оскільки підтримує реляційну модель даних, первинні та зовнішні ключі, індекси, унікальні обмеження і представлення. Це важливо для системи, у якій зберігаються взаємопов’язані дані про студентів, факультети, кафедри, наукові гуртки, заявки, заходи, результати діяльності та звітність.

Для забезпечення доступу користувачів використано механізм cookie-аутентифікації та рольову авторизацію. У системі передбачено ролі адміністратора, студента, керівника гуртка та представника деканату. Завдяки цьому кожна категорія користувачів отримує доступ лише до тих функцій, які відповідають її повноваженням.

Основні інструменти, використані для створення прикладного програмного забезпечення, наведено в таблиці 3.2.

Таблиця 3.2 – Інструментарій створення прикладного програмного забезпечення

Інструментарій	**Призначення у проєкті**
C#	Реалізація серверної логіки, моделей, контролерів, контексту бази даних і міграцій
ASP.NET Core	Платформа для створення вебзастосунку
ASP.NET Core MVC	Організація застосунку за моделлю “контролер – модель – представлення”
Razor Views	Формування вебсторінок інтерфейсу користувача
HTML	Структура вебсторінок
CSS	Оформлення зовнішнього вигляду сторінок
JavaScript	Реалізація клієнтської взаємодії
Bootstrap	Адаптивне оформлення інтерфейсу, форм, таблиць і кнопок
jQuery	Допоміжна клієнтська взаємодія на сторінках
jQuery Validation	Клієнтська перевірка правильності введення даних у формах
Entity Framework Core	Робота з базою даних через ORM
Microsoft SQL Server Express	Зберігання реляційних даних інформаційної системи
EF Core Migrations	Створення та оновлення структури бази даних
Cookie Authentication	Аутентифікація користувачів у системі

У файлі .csproj проєкту визначено використання Microsoft.NET.Sdk.Web, що підтверджує створення вебзастосунку на платформі .NET. Також у проєкті підключено NuGet-пакети Microsoft.EntityFrameworkCore.Design, Microsoft.EntityFrameworkCore.SqlServer та Microsoft.EntityFrameworkCore.Tools. Вони використовуються для роботи Entity Framework Core з Microsoft SQL Server, створення міграцій і підтримки структури бази даних.

На рисунку 3.2 наведено фрагмент файлу .csproj, який демонструє цільову платформу проєкту та підключені NuGet-пакети Entity Framework Core.

```
<Project Sdk="Microsoft.NET.Sdk.Web">

  <PropertyGroup>
    <TargetFramework>net10.0</TargetFramework>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
    <RootNamespace>Диплом_1</RootNamespace>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="Microsoft.EntityFrameworkCore.Design" Version="10.0.6">
      <PrivateAssets>all</PrivateAssets>
      <IncludeAssets>runtime; build; native; contentfiles; analyzers; buildtransitive</IncludeAssets>
    </PackageReference>
    <PackageReference Include="Microsoft.EntityFrameworkCore.SqlServer" Version="10.0.6" />
    <PackageReference Include="Microsoft.EntityFrameworkCore.Tools" Version="10.0.6">
      <PrivateAssets>all</PrivateAssets>
      <IncludeAssets>runtime; build; native; contentfiles; analyzers; buildtransitive</IncludeAssets>
    </PackageReference>
  </ItemGroup>

</Project>
```

Рисунок 3.2 – Фрагмент файлу проєкту з підключеними пакетами Entity Framework Core

На рисунку 3.3 наведено фрагмент файлу Program.cs, у якому показано підключення основних служб вебзастосунку, зокрема MVC, контексту бази даних, аутентифікації та авторизації.

```
builder.Services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme)
    .AddCookie(options =>
    {
        options.LoginPath = "/Account/Login";
        options.AccessDeniedPath = "/Account/AccessDenied";
        options.ExpireTimeSpan = TimeSpan.FromHours(8);
    });

builder.Services.AddControllersWithViews(options =>
{
    var policy = new AuthorizationPolicyBuilder()
        .RequireAuthenticatedUser()
        .Build();

    options.Filters.Add(new AuthorizeFilter(policy));
});

builder.Services.AddDbContext<StudentScienceClubsContext>(options =>
    options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));
```

Рисунок 3.3 – Налаштування основних служб вебзастосунку у файлі Program.cs

Таким чином, обраний інструментарій є доцільним для створення інформаційної системи обліку та аналізу діяльності студентських наукових гуртків. Використання C#, ASP.NET Core MVC, Razor Views, Entity Framework Core і Microsoft SQL Server Express забезпечує надійну серверну логіку, зручний вебінтерфейс, структуровану роботу з базою даних, підтримку ролей користувачів і можливість подальшого розширення програмного забезпечення.

3.3 Алгоритмізація та програмування програмних модулів

Програмне забезпечення інформаційної системи обліку та аналізу діяльності студентських наукових гуртків реалізовано у вигляді набору взаємопов’язаних програмних модулів. Кожен модуль відповідає за окрему функціональну ділянку системи та реалізований засобами ASP.NET Core MVC. Основна прикладна логіка зосереджена в контролерах, які взаємодіють із моделями предметної області, представленнями Razor Views і контекстом бази даних StudentScienceClubsContext.

У системі реалізовано модулі авторизації користувачів, обліку студентів, ведення наукових гуртків, обробки заявок на вступ до гуртків, формування членства, обліку засідань, відвідуваності, наукових заходів, результатів участі, аналітики та звітності. Такий підхід забезпечує модульність, логічну відокремленість функцій і спрощує подальший супровід програмного забезпечення.

Основні програмні модулі системи наведено в таблиці 3.3.

Таблиця 3.3 – Основні програмні модулі системи

Програмний модуль	Контролер	Основні таблиці БД	Основні функції	Ролі доступу
Авторизація та реєстрація	AccountController	AppUsers, Students, StudyGroups, AppLoginLogs	вхід, реєстрація, вихід, фіксація входів	неавторизовані, авторизовані
Облік студентів	StudentsController	Students, StudyGroups, AppUsers	перегляд, додавання, редагування, видалення студентів	Admin
Облік наукових гуртків	ScienceClubsController	ScienceClubs, Departments, Supervisors	перегляд, створення, редагування, видалення гуртків	Admin, авторизовані
Заявки на вступ	ClubApplicationsController	ClubApplications, ScienceClubs, Students, ClubMemberships	подання, скасування, погодження, відхилення заявок	Student, Supervisor, Admin
Членство в гуртках	ClubMembershipsController	ClubMemberships, MembershipStatuses	ведення складу учасників гуртків	Admin
Засідання гуртків	ClubMeetingsController	ClubMeetings, ScienceClubs	створення, редагування, видалення засідань	Admin
Облік відвідуваності	AttendancesController	Attendances, ClubMeetings, ClubMemberships	відмітка присутності, перегляд власної відвідуваності	Admin, Supervisor, Student
Наукові заходи	ScientificEventsController	ScientificEvents, EventTypes, EventStatuses	облік заходів, перегляд, редагування	Admin, авторизовані
Учасники заходів	EventParticipantsController	EventParticipants, ParticipationRoles	формування списку учасників заходу	Admin
Результати участі	ResultsController	ParticipationResults, EventParticipants	додавання, перегляд, редагування результатів	Admin, перегляд – авторизовані
Аналітика та звітність	AnalyticsController	ScienceClubs, ClubMemberships, ScientificEvents, ClubMeetings, ParticipationResults, ReportApprovals	формування звітів, експорт CSV, затвердження	Admin, Management
Журнал входів	LoginLogsController	AppLoginLogs	перегляд і пошук історії входів	Admin

Алгоритм авторизації користувача

Алгоритм авторизації реалізовано в контролері AccountController, зокрема в методі Login. Його призначення полягає у перевірці введених облікових даних, підтвердженні дійсності облікового запису, створенні автентифікованої сесії

користувача та перенаправленні до відповідного розділу системи залежно від ролі.

На першому етапі користувач вводить логін і пароль. Система виконує перевірку коректності моделі введення, після чого здійснює пошук активного користувача у таблиці AppUsers. Якщо користувача не знайдено або модель містить помилки, у журналі входів фіксується невдала спроба, а користувачу відображається повідомлення про помилку. Якщо користувача знайдено, система перевіряє правильність пароля. У разі неправильного пароля також виконується запис невдалого входу.

На рисунку 3.4 наведено блок-схему перевірки облікових даних користувача.

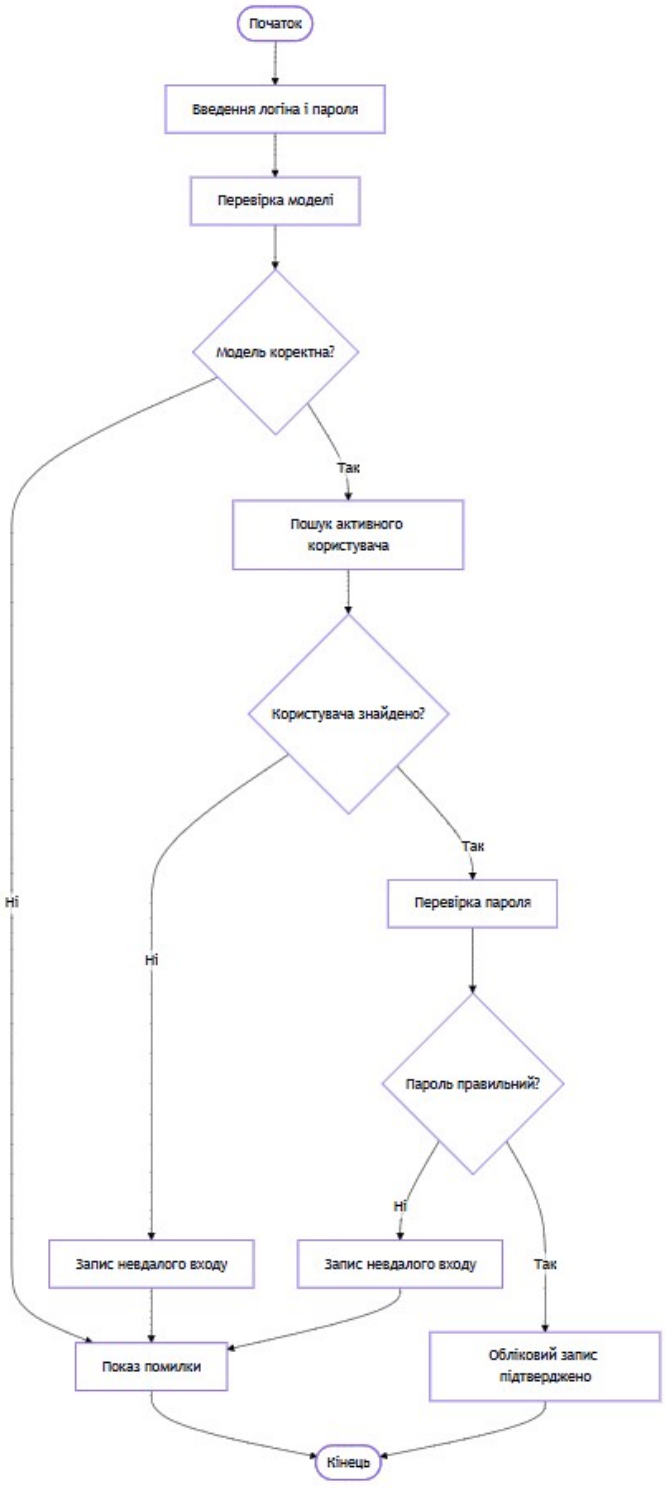


Рисунок 3.4 – Блок-схема перевірки облікових даних користувача

Після успішної перевірки пароля обліковий запис вважається підтвердженим. На наступному етапі формується набір claims, створюється cookie-аутентифікація, а факт успішного входу записується до таблиці AppLoginLogs. Після цього система визначає роль користувача та виконує

перенаправлення: студент переходить до кабінету студента, керівник – до кабінету керівника, представник деканату – до аналітичного розділу, адміністратор – на головну сторінку системи.

На рисунку 3.5 наведено блок-схему створення автентифікації та перенаправлення користувача за роллю.

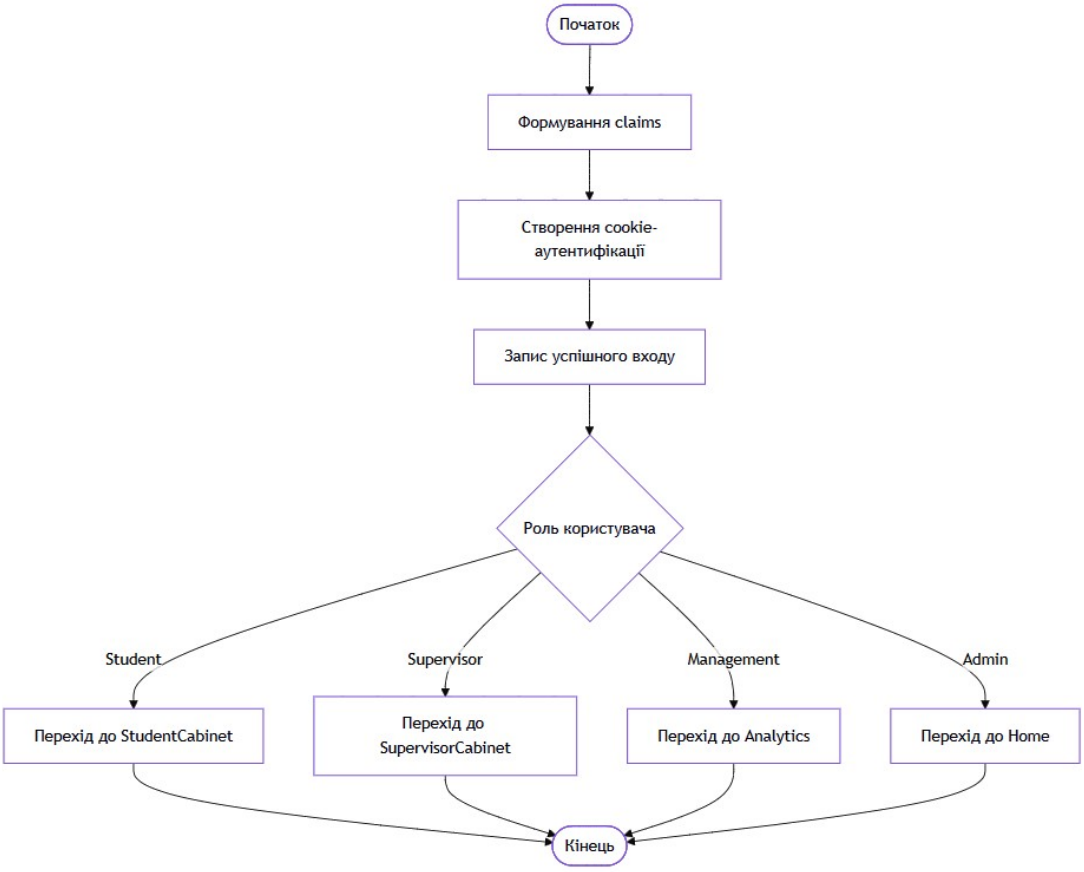


Рисунок 3.5 – Блок-схема створення автентифікації та перенаправлення користувача за роллю

Таким чином, алгоритм авторизації забезпечує не лише перевірку особи користувача, але й реєстрацію подій входу та розмежування доступу відповідно до ролі.

Алгоритм подання заявки студентом на вступ до наукового гуртка

Подання заявки на вступ до гуртка реалізовано в контролері ClubApplicationsController, зокрема в методі Apply. Цей алгоритм забезпечує контроль коректності подання заявки та унеможливлює дублювання записів.

Спочатку студент обирає науковий гурток. Після цього система перевіряє, чи авторизований студент у системі. Далі перевіряється, чи існує вибраний гурток і чи він є активним. Якщо хоча б одна з цих умов не виконується, користувачу забороняється подальше виконання операції або відображається повідомлення про помилку. Наступним кроком є перевірка членства студента в обраному гуртку. Якщо студент уже є учасником гуртка, повторне подання заявки не допускається. Якщо студент не є учасником, система перевіряє, чи існує в таблиці ClubApplications уже створена заявка зі статусом Pending. Якщо така заявка є, новий запис не створюється, а користувач отримує повідомлення про дублювання. Якщо всі перевірки пройдено успішно, система створює новий запис у таблиці ClubApplications, присвоює заявці статус Pending, зберігає її в базі даних і повідомляє студента про успішне подання.

На рисунку 3.6 наведено блок-схему алгоритму подання заявки студентом на вступ до наукового гуртка.

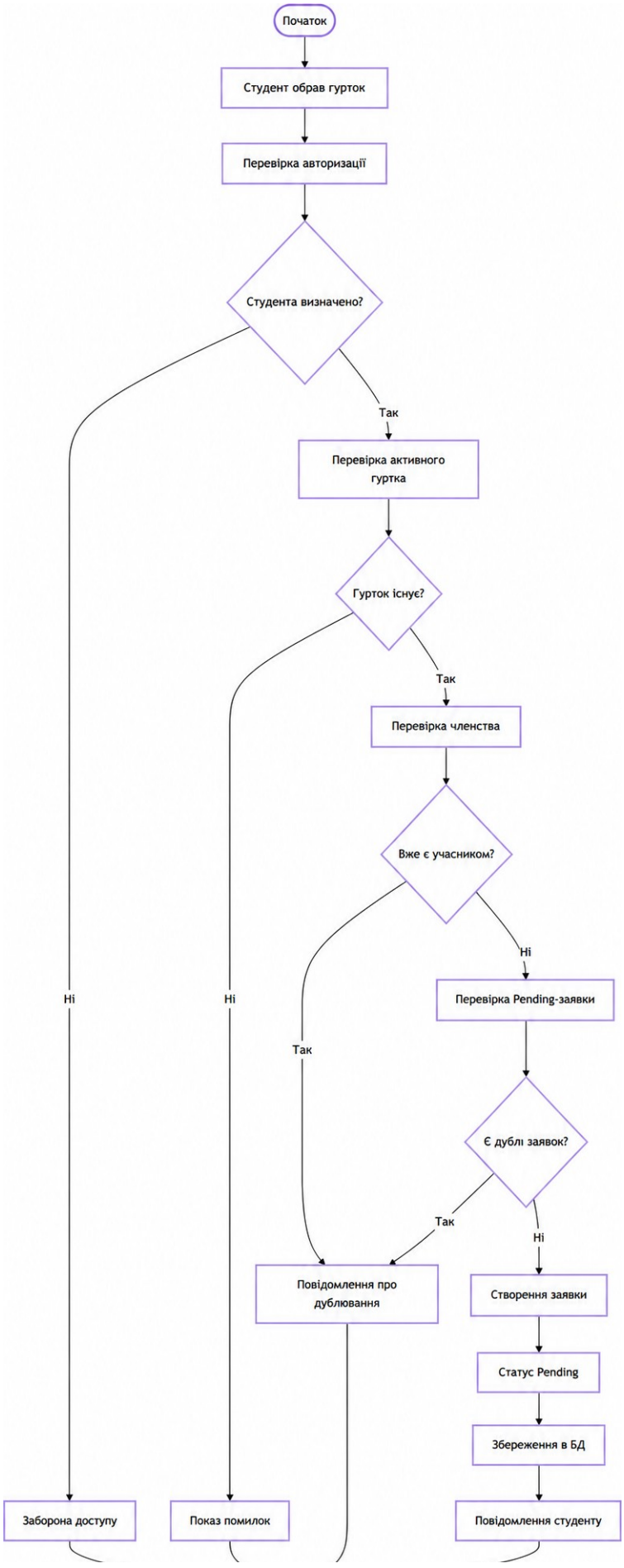

Початок
Студент обрав гурток
Перевірка авторизації
Студента визначено?
Так
Ні
Перевірка активного гуртка
Гурток існує?
Так
Ні
Перевірка членства
Вже є учасником?
Так
Ні
Перевірка Pending-заявки
Є дублі заявок?
Так
Ні
Повідомлення про дублювання
Створення заявки
Статус Pending
Збереження в БД
Повідомлення студенту
Заборона доступу
Показ помилок

Рисунок 3.6 – Блок-схема алгоритму подання заявки студентом на вступ до наукового гуртка

Отже, модуль подання заявки забезпечує контроль повторних звернень і підтримує цілісність даних щодо членства у гуртках.

Алгоритм розгляду заявки керівником гуртка або адміністратором

Розгляд заявки реалізовано в методах Approve і Reject контролера ClubApplicationsController. Алгоритм призначений для перевірки прав доступу, ухвалення рішення щодо заявки та внесення відповідних змін до бази даних. Спочатку користувач відкриває список заявок і обирає конкретну заявку для розгляду. Система перевіряє, чи має поточний користувач право виконувати розгляд. Якщо право не підтверджено, доступ до операції забороняється.

У разі підтвердження права користувач ухвалює рішення: схвалити або відхилити заявку. Якщо заявка схвалюється, система спочатку перевіряє наявність статусу ACTIVE, а потім перевіряє, чи не існує вже чинного членства студента у відповідному гуртку. Якщо членства немає, створюється запис у таблиці ClubMemberships, а статус заявки змінюється на Approved. Якщо ж заявка відхиляється, її статус змінюється на Rejected.

Після цього система зберігає коментар до розгляду, дату ухвалення рішення та користувача, який виконав розгляд. Далі оновлені дані записуються до бази.

На рисунку 3.7 наведено блок-схему алгоритму розгляду заявки керівником гуртка або адміністратором.

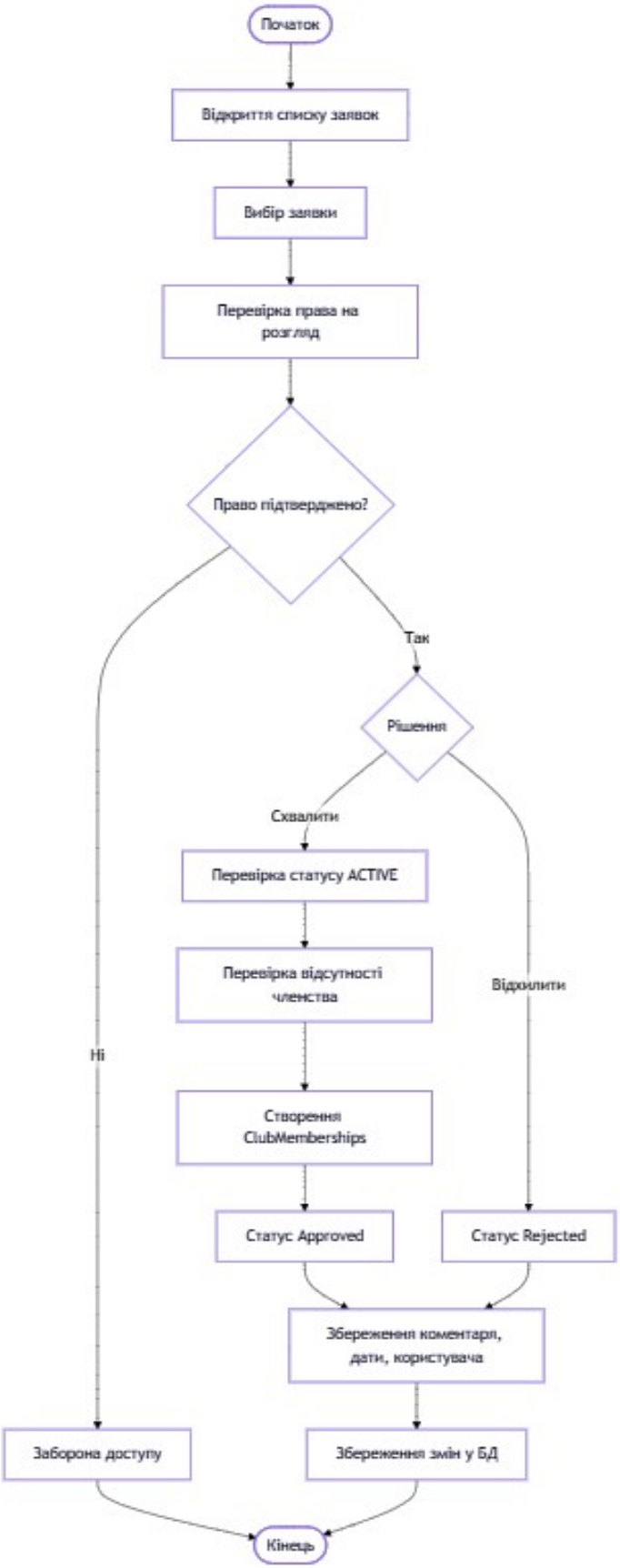


Рисунок 3.7 – Блок-схема алгоритму розгляду заявки керівником гуртка або адміністратором

У результаті виконання цього алгоритму система забезпечує контрольований процес переходу заявки зі стану очікування до стану схвалення або відхилення.

Алгоритм обліку відвідуваності учасників гуртка

Модуль обліку відвідуваності реалізовано в контролері AttendancesController, зокрема в методах Index і Save. Його основне призначення полягає у формуванні списку учасників обраного засідання гуртка та збереженні відміток про присутність або відсутність.

На початковому етапі адміністратор або керівник гуртка обирає засідання. Система формує перелік доступних засідань: для адміністратора доступні всі записи, а для керівника – лише засідання тих гуртків, за які він відповідає. Після вибору засідання система отримує список активних учасників гуртка через таблицю ClubMemberships, а також зчитує вже існуючі відмітки з таблиці Attendances.

Після цього користувач встановлює ознаку присутності або відсутності для кожного учасника, за потреби додає причину відсутності. Під час збереження система для кожного студента або створює новий запис, або оновлює вже існуючий. При цьому фіксуються дата відмітки та користувач, який виконав дію.

Таким чином, алгоритм забезпечує ведення обліку відвідуваності засідань гуртка та дозволяє зберігати актуальні дані щодо активності учасників.

Алгоритм додавання результатів участі у наукових заходах

Модуль обліку результатів участі реалізовано в контролері ResultsController, насамперед у методі Create. Його призначення полягає у фіксації досягнень учасників наукових заходів.

На початковому етапі адміністратор обирає учасника заходу, після чого вводить характеристики результату: тип результату, назву, оцінку, відомості про нагороду, посилання на публікацію та примітки. Після введення даних система перевіряє, чи належить значення оцінки допустимому діапазону від 0 до 100.

Додатково виконується перевірка на дублювання результату для того самого учасника. Якщо запис із такими самими параметрами вже існує, система не створює дубльований запис. Якщо ж перевірка успішна, відомості зберігаються у таблиці ParticipationResults.

Отже, алгоритм дозволяє систематизовано зберігати результати участі студентів у наукових заходах та контролювати коректність внесених даних.

Алгоритм формування аналітичного звіту

Модуль аналітики реалізовано в контролері AnalyticsController, зокрема в методах Index, BuildReportRowsAsync, ExportCsv і ApproveReport. Його призначення полягає у формуванні узагальненого звіту про діяльність студентських наукових гуртків за вибраними критеріями.

На першому етапі користувач відкриває сторінку аналітики, після чого обирає факультет, гурток і часовий період. Система перевіряє коректність введеного діапазону дат. Якщо дати некоректні, користувачу відображається повідомлення про помилку. У разі успішної перевірки система формує список гуртків і для кожного з них послідовно виконує підрахунок кількості учасників, заходів, засідань і результатів участі. Після цього формується об’єкт AnalyticsReportRow, на основі якого будується модель звіту та відображається результат на екрані.

На рисунку 3.8 наведено блок-схему алгоритму формування аналітичного звіту.

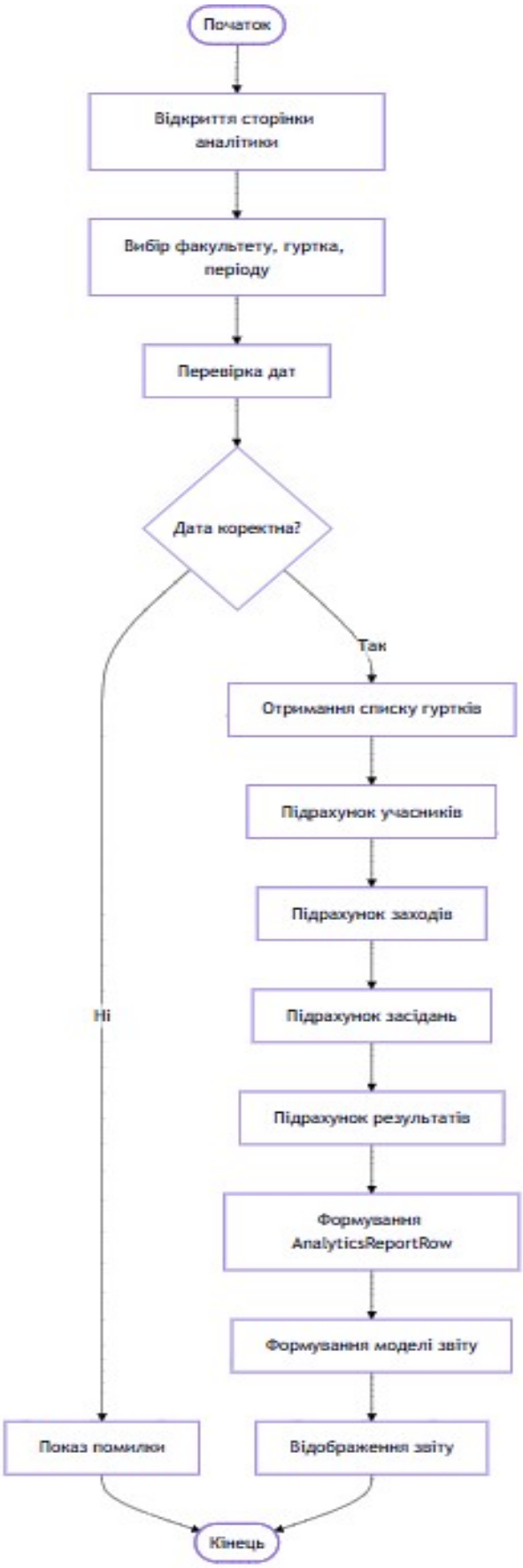


Рисунок 3.8 – Блок-схема алгоритму формування аналітичного звіту

Після формування звіту система може виконувати додаткові дії. Зокрема, якщо користувач ініціює експорт, формується CSV-файл і передається користувачу. Якщо ж потрібне офіційне затвердження звіту, система створює запис у таблиці ReportApprovals, зберігає основні показники звіту, коментар користувача та записує результати до бази даних.

На рисунку 3.9 наведено блок-схему експорту та затвердження аналітичного звіту.

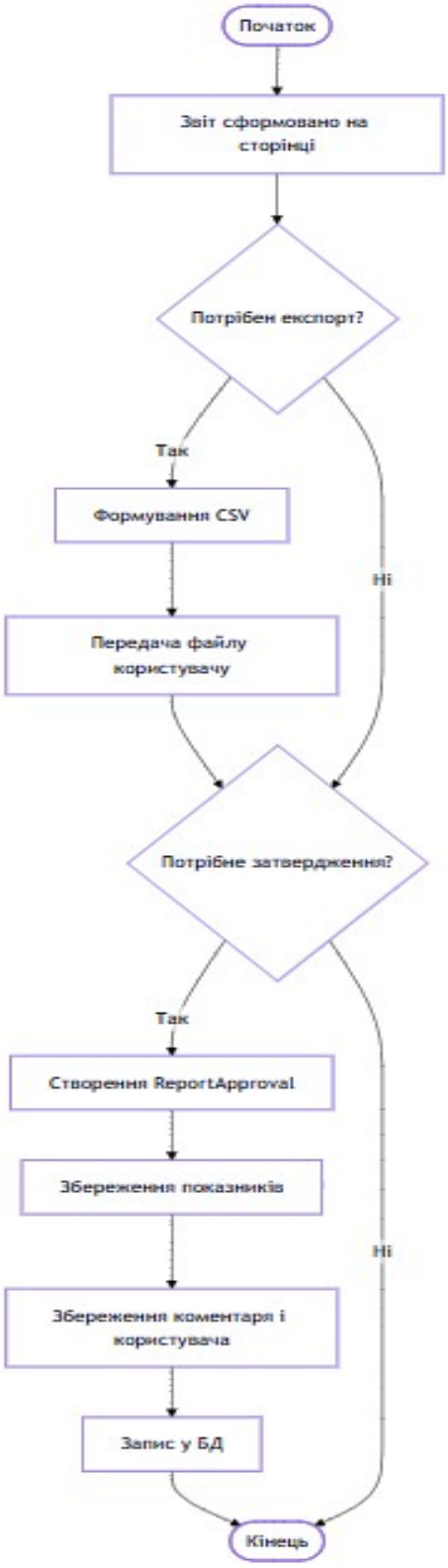

Початок
Звіт сформовано на сторінці
Потрібен експорт?
Так
Ні
Формування CSV
Передача файлу користувачу
Потрібне затвердження?
Так
Ні
Створення ReportApproval
Збереження показників
Збереження коментаря і користувача
Запис у БД
Кінець

Рисунок 3.9 – Блок-схема експорту та затвердження аналітичного звіту

Таким чином, модуль аналітики забезпечує як формування зведеної інформації про діяльність гуртків, так і подальше збереження або експорт отриманих результатів.

Рекомендовані фрагменти коду для ілюстрацій

Для підтвердження практичної реалізації описаних алгоритмів доцільно використати фрагменти коду відповідних контролерів. Рекомендовані фрагменти наведено в таблиці 3.4.

Таблиця 3.4 – Рекомендовані фрагменти коду для ілюстрацій

Файл	Метод	Що демонструє
AccountController.cs	Login	перевірка користувача, пароля, створення cookie-аутентифікації та перенаправлення за роллю
ClubApplicationsController.cs	Apply	перевірка дублюючої заявки, перевірка членства та створення запису в ClubApplications
ClubApplicationsController.cs	Approve / Reject	перевірка права на розгляд заявки, зміна статусу та створення членства
AttendancesController.cs	Save	збереження відміток присутності, причин відсутності та користувача, який виконав відмітку
AnalyticsController.cs	BuildReportRowsAsync	підрахунок кількості учасників, заходів, засідань і результатів для формування звіту
AccountController.cs	LogLoginAsync	фіксація успішних і невдалих входів у таблиці AppLoginLogs

Висновок до підрозділу 3.3

У підрозділі було розглянуто алгоритмізацію та програмну реалізацію основних функціональних модулів інформаційної системи обліку та аналізу діяльності студентських наукових гуртків. Реалізовані модулі охоплюють

процеси авторизації користувачів, подання та розгляду заявок, ведення відвідуваності, обліку результатів участі та формування аналітичних звітів. Побудовані блок-схеми наочно відображають послідовність виконання основних операцій у системі, а програмна реалізація в контролерах ASP.NET Core MVC забезпечує практичне виконання описаних алгоритмів. Це підтверджує цілісність прикладного програмного забезпечення та його відповідність функціональним вимогам інформаційної системи.

4 РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ТА ЕКСПЛУАТАЦІЇ СИСТЕМИ

Розроблена інформаційна система обліку та аналізу діяльності студентських наукових гуртків є вебзастосунком, реалізованим на платформі ASP.NET Core MVC. Система працює з реляційною базою даних Microsoft SQL Server Express, а доступ до даних здійснюється через Entity Framework Core та контекст бази даних StudentScienceClubsContext.

У цьому розділі наведено рекомендації щодо впровадження та експлуатації системи, результати тестування основних функціональних модулів, вимоги до апаратного й програмного забезпечення, а також склад інсталяційного пакету для локального розгортання або розміщення системи на хостингу.

4.1 Тестування системи

Тестування розробленої інформаційної системи обліку та аналізу діяльності студентських наукових гуртків проводиться з метою перевірки працездатності її основних функціональних модулів, коректності обробки даних, роботи механізмів авторизації та розмежування доступу за ролями, взаємодії з базою даних і формування аналітичної та звітної інформації.

Окремого тестового проєкту або набору автоматичних unit-тестів у структурі рішення не передбачено, тому перевірка функціонування системи здійснюється вручну за підготовленими тестовими сценаріями. Такий підхід дозволяє перевірити роботу системи з позиції кінцевих користувачів і підтвердити коректність виконання основних бізнес-процесів.

До об’єктів тестування належать модулі авторизації та реєстрації користувачів, рольового доступу, подання заявок студентами, розгляду заявок керівниками гуртків або адміністраторами, створення членства в гуртках, ведення відвідуваності, внесення результатів участі, формування аналітичних звітів, експорту CSV, затвердження звітів та журналу входів.

Таблиця 4.1 – Види тестування системи

Вид тестування	Мета	Що перевіряється
Функціональне тестування	Перевірка коректності роботи основних модулів	заявки, відвідуваність, результати, аналітика
Тестування авторизації	Перевірка входу і доступу за ролями	логін, перенаправлення за ролями, відмова в доступі
Тестування валідації даних	Перевірка обробки коректних і помилкових значень	пароль, оцінка, дубль заявки, діапазон дат
Тестування БД	Перевірка збереження та читання даних	AppUsers, ClubApplications, Attendances, ParticipationResults, ReportApprovals
Тестування звітності	Перевірка формування, експорту і затвердження звітів	аналітика, CSV, затвердження звіту
Тестування інтерфейсу	Перевірка доступності сторінок і навігації	форми, таблиці, повідомлення, кабінети
Тестування помилкових сценаріїв	Перевірка реакції системи на некоректні дії	невірний пароль, повторна заявка, некоректні дати

Для перевірки працездатності системи сформовано набір тестових сценаріїв, які охоплюють основні дії користувачів і типові помилкові ситуації. Основну увагу приділено авторизації, роботі ролей, поданню та розгляду заявок, збереженню відвідуваності, внесенню результатів участі, формуванню звітів і фіксації подій входу.

Таблиця 4.2 – Тестові сценарії перевірки роботи системи

№	Модуль	Тестова дія	Очікуваний результат	Фактичний результат	Статус
1	Авторизація	Вхід з правильним логіном і паролем	Успішний вхід у систему	Вхід виконується успішно	Пройдено
2	Авторизація	Вхід з неправильним паролем	Показ помилки, запис невдалого входу	Помилка відображається, запис у журналі формується	Пройдено
3	Ролі	Вхід студента	Перенаправлення до StudentCabinet	Перенаправлення виконується	Пройдено
4	Ролі	Вхід керівника	Перенаправлення до SupervisorCabinet	Перенаправлення виконується	Пройдено
5	Ролі	Вхід Management	Перенаправлення до Analytics	Перенаправлення виконується	Пройдено
6	Заявки	Подання	Створення	Заявка	Пройден

		заявки студентом	заявки зі статусом Pending	зберігається	о
7	Заявки	Повторне подання Pending-заявки	Заборона дублювання	Система показує повідомлення про дубль	Пройдено
8	Заявки	Подання заявки чинним учасником гуртка	Заборона повторного вступу	Система блокує подання	Пройдено
9	Заявки	Схвалення заявки	Статус заявки змінюється на Approved	Статус оновлюється	Пройдено
10	Членство	Схвалення заявки без чинного членства	Створення запису в ClubMemberships	Запис створюється	Пройдено
11	Заявки	Відхилення заявки	Статус заявки змінюється на Rejected	Статус оновлюється	Пройдено
12	Відвідуваність	Збереження відміток	Створення або оновлення записів у Attendances	Дані зберігаються	Пройдено
13	Відвідуваність	Перегляд студентом власної відвідуваності	Відображення сторінки власної відвідуваності	Дані відображаються	Пройдено
14	Результати	Додавання результату участі	Створення запису в ParticipationResults	Запис створюється	Пройдено
15	Результати	Введення оцінки поза межами 0–100	Повідомлення про помилку	Валідація спрацьовує	Пройдено
16	Аналітика	Формування звіту	Побудова аналітичних рядків	Звіт формується	Пройдено
17	Аналітика	Дата початку більша за дату завершення	Показ помилки	Помилка відображається	Пройдено
18	Звітність	Експорт звіту у CSV	Формування і передача CSV-файлу	CSV формується	Пройдено
19	Звітність	Затвердження звіту Management	Створення запису в ReportApprovals	Запис створюється	Пройдено
20	Журнал входів	Успішний або невдалий вхід	Запис події в AppLoginLogs	Журнал поповнюється	Пройдено

За результатами перевірки основних сценаріїв встановлено, що система коректно реалізує ключові функції предметної області. Модулі авторизації, рольового доступу, подання та розгляду заявок, ведення відвідуваності, внесення результатів участі, формування аналітичних звітів, експорту CSV і затвердження звітів функціонують відповідно до закладеної логіки.

Система правильно реагує на помилкові дії користувача: блокує вхід із неправильним паролем, не дозволяє повторне подання заявки зі статусом

Pending, не дозволяє подання заявки студентом, який уже є учасником гуртка, перевіряє межі оцінки результату та контролює коректність діапазону дат у звіті.

Таблиця 4.3 – Результати тестування основних модулів

Модуль	Перевірені функції	Підсумок
Авторизація	вхід, помилки входу, перенаправлення за ролями	Працює коректно
Заявки	подання, дублювання, схвалення, відхилення	Працює коректно
Членство	створення після схвалення заявки	Працює коректно
Відвідуваність	завантаження списку, збереження, перегляд студентом	Працює коректно
Результати участі	додавання, валідація оцінки, перевірка дублювання	Працює коректно
Аналітика	формування звіту, перевірка дат	Працює коректно
Звітність	експорт CSV, затвердження звіту	Працює коректно
Журнал входів	фіксація успішних і невдалих входів	Працює коректно

Під час перевірки модуля авторизації встановлено, що система підтримує вхід користувача за логіном і паролем, перевірку активності облікового запису та правильності пароля, створення cookie-аутентифікації, а також перенаправлення користувача на сторінку відповідно до його ролі.

У процесі тестування модуля заявок перевірено, що студент може подати заявку на вступ до гуртка, якщо він авторизований, гурток активний, а в системі відсутні дублюючі заявки зі статусом Pending або чинне членство в тому самому гуртку. При схваленні заявки система змінює її статус на Approved і створює новий запис у таблиці ClubMemberships, якщо чинного членства ще немає.

Під час тестування модуля відвідуваності перевірено вибір засідання, завантаження списку активних учасників гуртка, позначення присутності або відсутності та збереження відповідних записів у таблиці Attendances. Модуль аналітики перевірено шляхом формування звіту за вибраними параметрами, експорту звіту у CSV та затвердження звіту користувачем із роллю Management.

Таблиця 4.4 – Рекомендовані ілюстрації для підтвердження тестування

Позначення	Зміст ілюстрації
Рисунок 4.1	Сторінка входу до інформаційної системи
Рисунок 4.2	Кабінет студента після успішної авторизації
Рисунок 4.3	Подання заявки на вступ до наукового гуртка
Рисунок 4.4	Сторінка обліку відвідуваності учасників гуртка
Рисунок 4.5	Формування аналітичного звіту
Рисунок 4.6	Сторінка аналітики та експорту звіту у CSV

Для підтвердження результатів тестування нижче наведено приклади роботи основних модулів інформаційної системи.

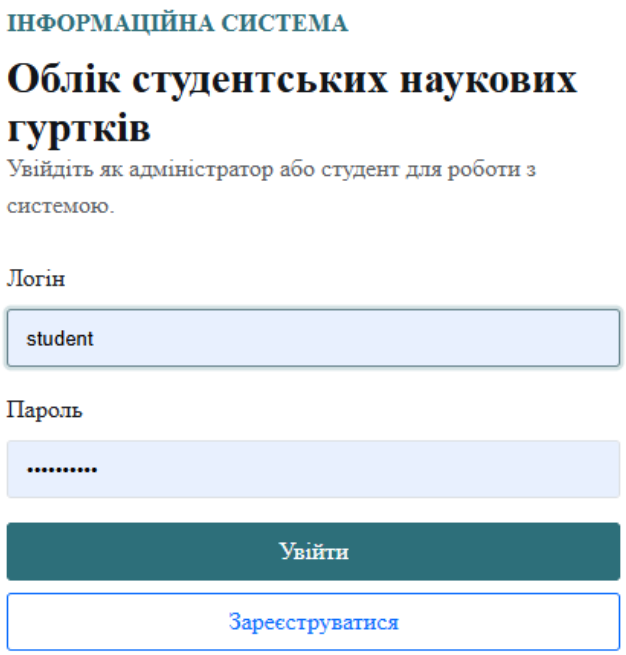


Рисунок 4.1 – Сторінка входу до інформаційної системи

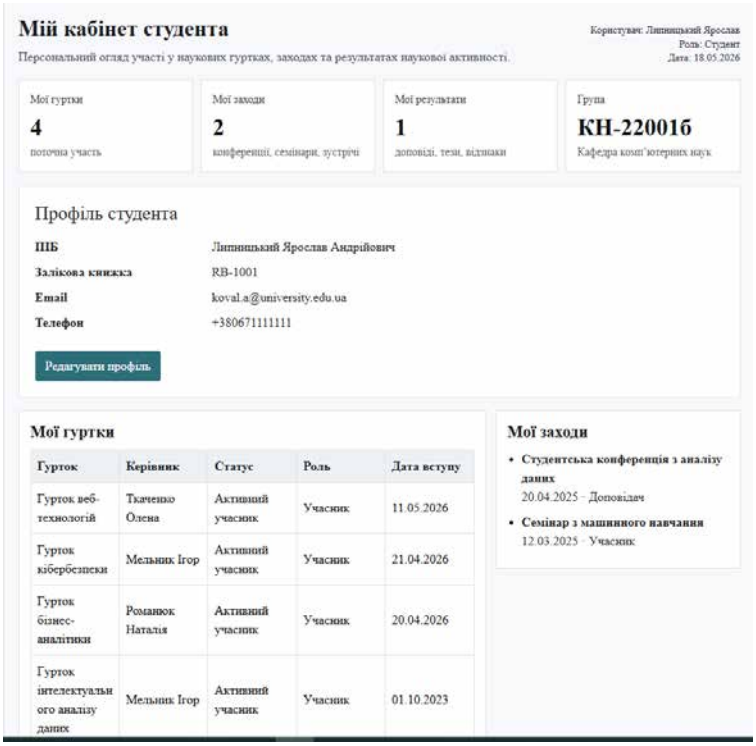


Рисунок 4.2 – Кабінет студента після успішної авторизації

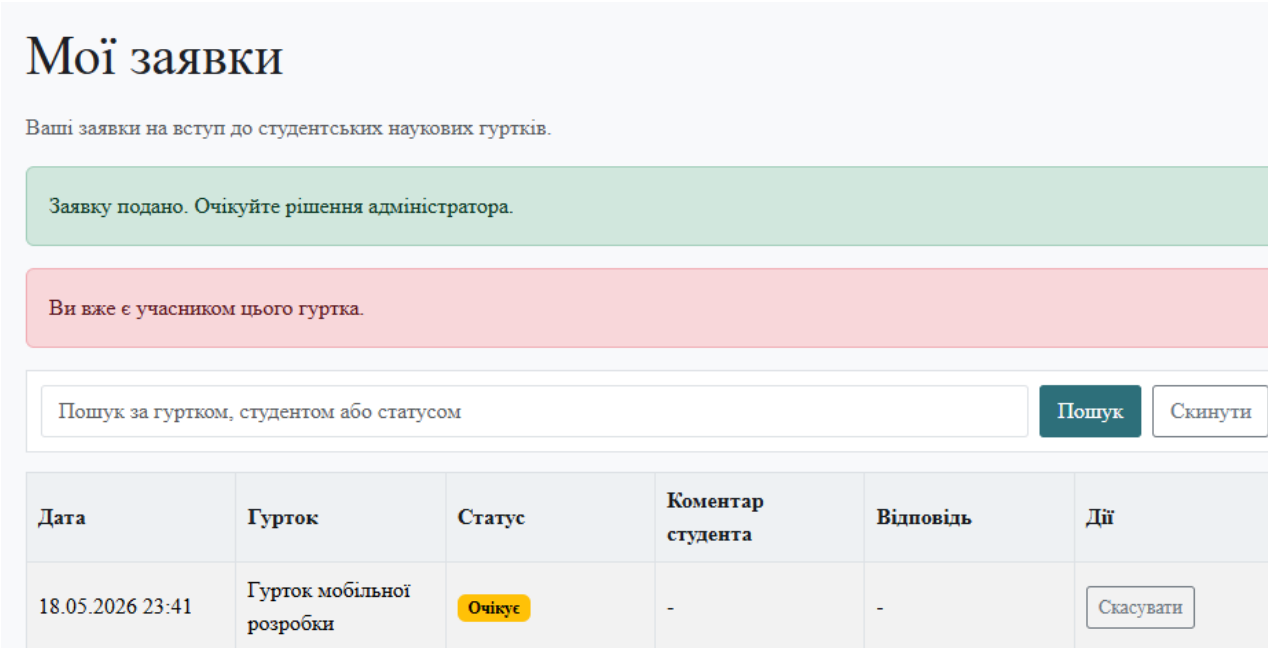


Рисунок 4.3 – Подання заявки на вступ до наукового гуртка

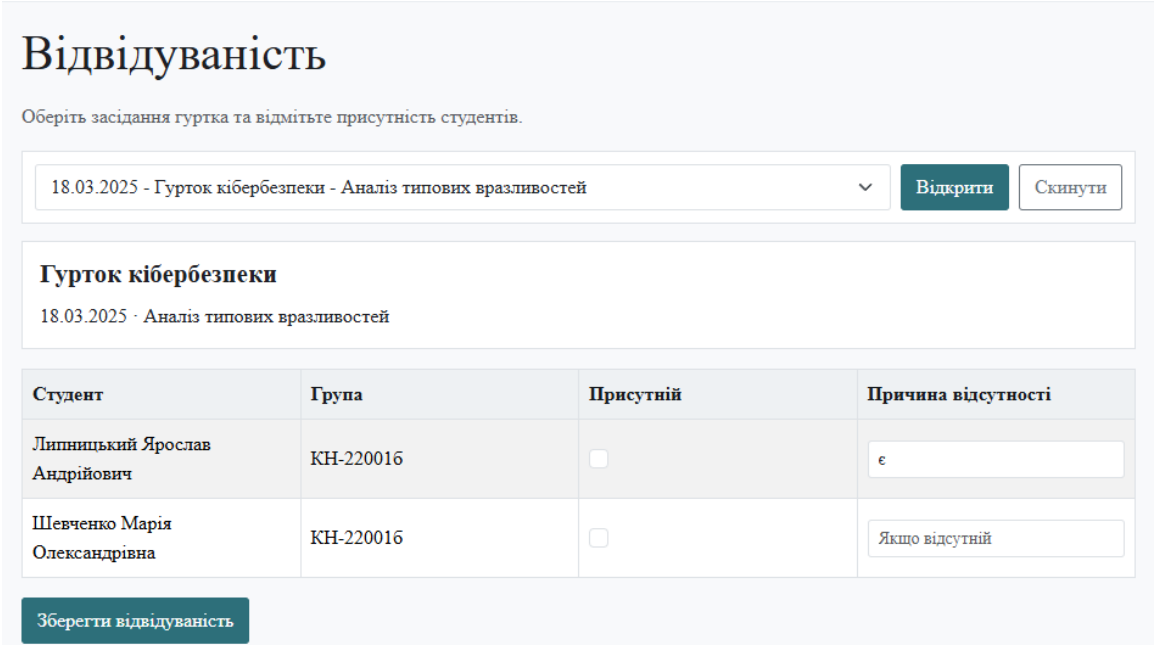


Рисунок 4.4 – Сторінка обліку відвідуваності учасників гуртка

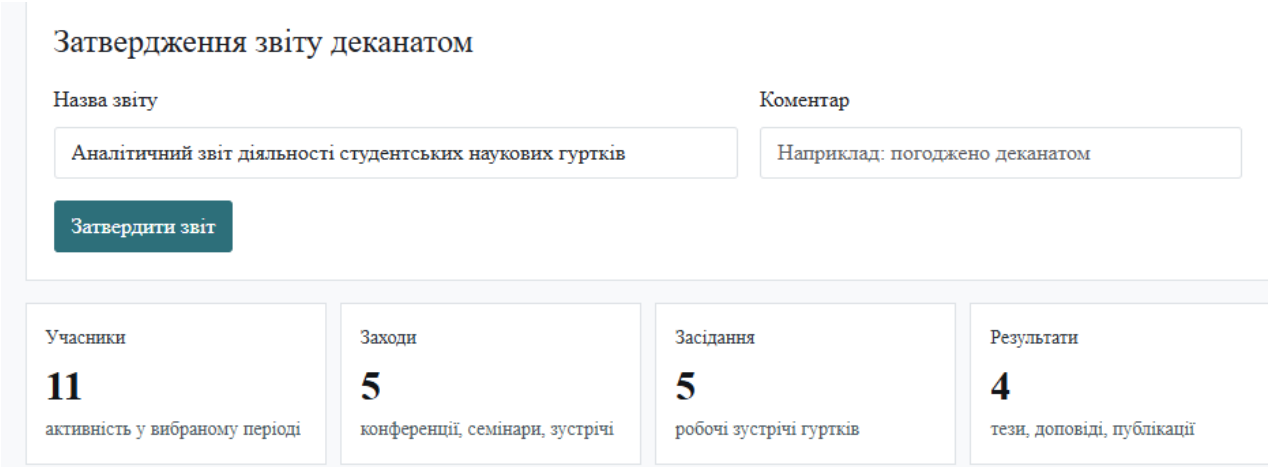


Рисунок 4.5 – Формування аналітичного звіту

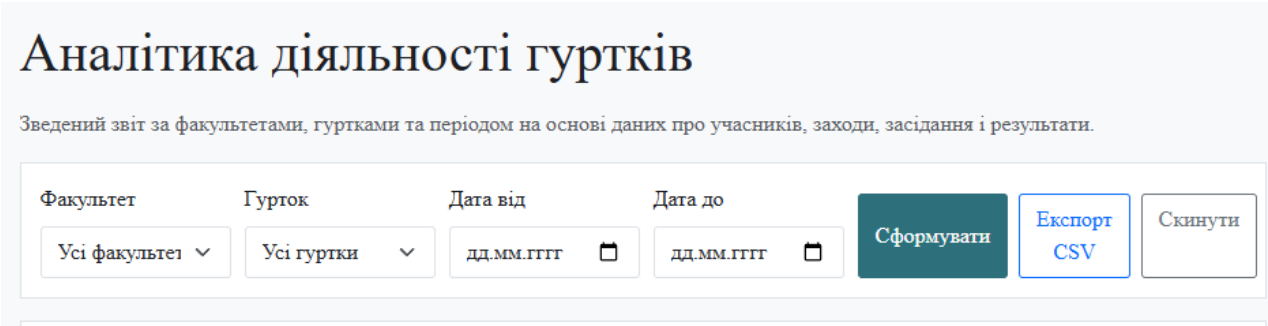
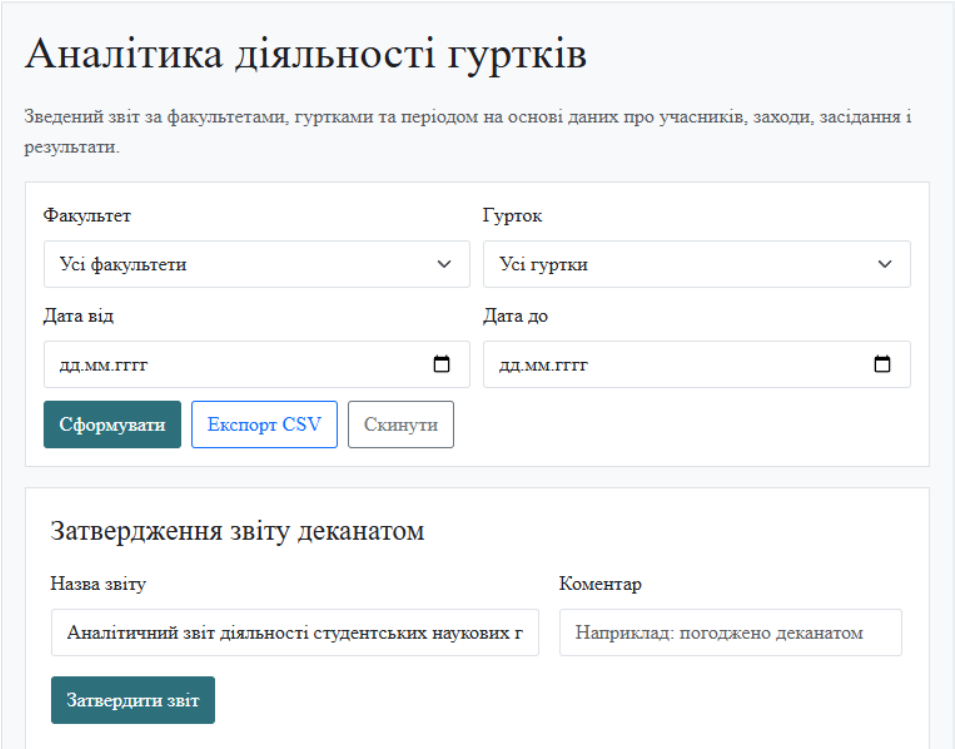


Рисунок 4.6 – Сторінка аналітики та експорту звіту у CSV



Висновок до підрозділу 4.1

Проведене тестування підтвердило працездатність основних модулів інформаційної системи обліку та аналізу діяльності студентських наукових гуртків. Система забезпечує коректну авторизацію користувачів, розмежування доступу за ролями, обробку заявок, ведення відвідуваності, внесення результатів участі та формування аналітичної звітності.

Реакція системи на помилкові сценарії також є коректною: при неправильних облікових даних відображається повідомлення про помилку, дублювання заявок блокується, некоректні значення оцінки не приймаються, а неправильний діапазон дат у звіті контролюється. Отже, система готова до подальшої тестової або навчальної експлуатації з можливістю подальшого розширення набору автоматизованих тестів.

4.2 Вимоги до апаратного та програмного забезпечення

Для стабільної роботи інформаційної системи необхідно визначити вимоги до апаратного та програмного забезпечення серверної і клієнтської частини. Це пов’язано з тим, що система реалізована як вебзастосунок на платформі

ASP.NET Core MVC, працює через браузер, використовує Entity Framework Core для взаємодії з базою даних і зберігає інформацію в Microsoft SQL Server Express.

Апаратні вимоги залежать від кількості користувачів, обсягу даних і способу розгортання системи. Для навчального або локального використання достатньо робочого комп'ютера з базовими характеристиками, однак для стабільної багатокористувацької експлуатації доцільно використовувати окремий комп'ютер або сервер.

Таблиця 4.5 – Мінімальні та рекомендовані вимоги до апаратного забезпечення

Параметр	Мінімальні вимоги	Рекомендовані вимоги
Процесор	2 ядра	4 ядра і більше
Оперативна пам'ять	4 ГБ	8–16 ГБ
Вільне місце на диску	від 10 ГБ	від 20–50 ГБ
Тип накопичувача	HDD або SSD	SSD
Мережеве підключення	локальне або базове мережеве підключення	стабільне підключення в локальній мережі
Тип пристрою	робочий комп'ютер	окремий сервер або виділений комп'ютер

До програмного забезпечення, необхідного для функціонування системи, належать операційна система Windows 10, Windows 11 або Windows Server, платформа .NET SDK або .NET Runtime, сумісна з цільовою платформою проєкту, система управління базами даних Microsoft SQL Server Express, а також ASP.NET Core MVC та Entity Framework Core.

Інтерфейс користувача побудований за допомогою Razor Views, тому робота з системою здійснюється через сучасний веббраузер. Для супроводу

системи та внесення змін до програмного коду доцільно використовувати Visual Studio або інше середовище, сумісне з розробкою на платформі .NET.

Таблиця 4.6 – Вимоги до програмного забезпечення

Компонент	Вимога
Операційна система	Windows 10 / Windows 11 / Windows Server
Платформа .NET	.NET SDK або .NET Runtime
Цільова платформа проєкту	відповідно до налаштувань проєкту .NET
Вебфреймворк	ASP.NET Core MVC
ORM	Entity Framework Core
СУБД	Microsoft SQL Server Express
Інтерфейс	Razor Views
Браузер	сучасний веббраузер
Середовище супроводу	Visual Studio або сумісне середовище
База даних	StudentScienceClubsDB

Серверна частина системи повинна забезпечувати запуск ASP.NET Core MVC застосунку, підключення до Microsoft SQL Server, виконання міграцій Entity Framework Core, обробку HTTP-запитів, авторизацію користувачів і роботу з базою даних через StudentScienceClubsContext.

Клієнтська частина не потребує встановлення окремого програмного забезпечення. Користувачу достатньо мати комп’ютер, ноутбук або інший пристрій із сучасним браузером, доступом до локальної мережі або вебадреси системи та обліковим записом із правами доступу відповідно до ролі користувача.

Таблиця 4.7 – Вимоги до серверної та клієнтської частини

Частина системи	Основні вимоги
Серверна частина	запуск ASP.NET Core MVC застосунку, доступ до SQL Server Express, виконання EF Core Migrations, обробка HTTP-запитів,

	авторизація, робота з БД
Клієнтська частина	пристрій користувача, сучасний браузер, доступ до локальної мережі або вебадреси, права доступу за роллю
База даних	наявність StudentScienceClubsDB, підтримка таблиць, ключів, зв’язків, індексів, представлень, резервне копіювання

Експлуатація системи залежить від ролі користувача. Адміністратор контролює коректність основних даних системи, студент подає заявки та переглядає власну інформацію, керівник гуртка розглядає заявки й веде відвідуваність, а представник деканату / Management працює з аналітичними звітами, їх експортом і затвердженням.

Під час адміністрування системи необхідно регулярно перевіряти облікові записи користувачів, контролювати їх активність, коректність ролей і відповідність користувачів реальним студентам або керівникам гуртків. Особливу увагу слід приділяти обліковим записам із розширеними правами доступу, зокрема ролям Admin і Management. Для контролю використання системи доцільно переглядати журнал входів AppLoginLogs. Він дозволяє відстежувати успішні та невдалі спроби входу, дату авторизації, ім’я користувача та інші службові дані. Це підвищує рівень контролю за доступом до системи та допомагає своєчасно виявляти підозрілу активність.

У системі використовуються механізми рольової авторизації, cookie-аутентифікації та хешування паролів. Доступ до окремих модулів обмежується відповідно до ролі користувача, що дозволяє зменшити ризик несанкціонованого перегляду або зміни даних.

База даних StudentScienceClubsDB повинна регулярно резервуватися стандартними засобами Microsoft SQL Server. За активної експлуатації системи доцільно виконувати щоденне резервне копіювання, а за невеликої інтенсивності використання — щотижневе. Копії бажано зберігати на окремому носії або сервері та періодично перевіряти можливість їх відновлення.

Таким чином, визначені вимоги до апаратного та програмного забезпечення забезпечують можливість локального розгортання, тестування та подальшої експлуатації інформаційної системи. Запропоновані характеристики є достатніми для стабільної роботи вебзастосунку, коректної взаємодії з базою даних і доступу користувачів до основних функціональних модулів системи.

4.3 Склад інсталяційного пакету

Інсталяційний пакет інформаційної системи обліку та аналізу діяльності студентських наукових гуртків призначений для розгортання вебзастосунку на сервері або хостингу. З урахуванням реальної структури проєкту пакет повинен містити скомпільовані файли ASP.NET Core MVC застосунку, конфігураційні файли, статичні ресурси, міграції Entity Framework Core або підготовлені засоби створення бази даних, а також супровідну документацію для встановлення і первинного налаштування системи.

Оскільки система є вебзастосунком, склад інсталяційного пакету має забезпечувати не лише запуск програмної частини, а й правильне підключення до бази даних Microsoft SQL Server, налаштування середовища виконання та доступ користувачів через браузер.

Таблиця 4.8 – Склад інсталяційного пакету

№	Компонент	Призначення
1	Publish-файли ASP.NET Core MVC застосунку	Запуск вебзастосунку на сервері або хостингу
2	Виконувальні файли та бібліотеки .NET	Забезпечення роботи застосунку після публікації
3	Конфігураційні файли	Налаштування параметрів застосунку та середовища
4	Папка wwwroot	Зберігання статичних ресурсів інтерфейсу
5	Міграції EF Core	Створення та оновлення структури бази даних
6	SQL-скрипт або резервна копія БД	Початкове розгортання або відновлення бази даних
7	Інструкція з розгортання	Опис дій адміністратора під час

		встановлення
8	Інструкція з налаштування БД	Налаштування доступу до Microsoft SQL Server
9	Опис ролей користувачів	Первинне ознайомлення з доступом до системи

До складу інсталяційного пакету доцільно включати publish-файли застосунку, виконувальні бібліотеки .NET, файл запуску застосунку, конфігураційні файли appsettings.json та, за потреби, окремий файл production-конфігурації. Важливим елементом пакету є папка wwwroot, яка містить CSS, JavaScript та підключені клієнтські бібліотеки.

Таблиця 4.9 – Файли та каталоги, що входять до пакету розгортання

Файл / каталог	**Наявність**	**Призначення**
Program.cs	Так	Конфігурація запуску застосунку
appsettings.json	Так	Базова конфігурація застосунку
appsettings.Production.json	За потреби	Може використовуватися для production-середовища
wwwroot	Так	Статичні файли інтерфейсу
Views	Так	Razor-представлення вебінтерфейсу
Data/ StudentScienceClubsContext.cs	Так	Контекст бази даних
Migrations	Так	Міграції Entity Framework Core
.csproj	Так	Параметри проєкту та підключені пакети
web.config	За потреби	Може бути сформований під час publish для IIS

До інсталяційного пакету не слід включати паролі, відкриті connection string із приватними даними, персональні дані користувачів без необхідності, тимчасові файли, локальні службові файли середовища розробника, а також непотрібні файли Visual Studio. Якщо використовується готовий publish-пакет, до нього не потрібно додавати вихідні папки bin та obj із робочого проєкту, оскільки вони не є частиною кінцевого пакету розгортання.

Формування інсталяційного пакету має виконуватися поетапно. Спочатку необхідно перевірити роботу системи локально, підготувати конфігурацію для production-середовища, виконати публікацію застосунку, перевірити наявність статичних файлів, підготувати засоби створення або відновлення бази даних, додати супровідну документацію та виконати тестовий запуск пакету.

Таблиця 4.10 – Етапи формування інсталяційного пакету

№	Етап	Зміст
1	Локальна перевірка	Перевірка працездатності системи у середовищі розробника
2	Налаштування production	Підготовка конфігурації для розгортання
3	Публікація застосунку	Виконання publish через dotnet publish або Visual Studio
4	Підготовка бази даних	Підготовка міграцій, SQL-скрипту або резервної копії
5	Перевірка статичних файлів	Контроль наявності ресурсів папки wwwroot
6	Додавання документації	Підготовка інструкцій для розгортання та користувачів
7	Тестовий запуск	Перевірка запуску пакету в тестовому середовищі

Розгортання системи на хостингу передбачає підготовку сервера, встановлення або наявність .NET Runtime, створення бази даних SQL Server, завантаження publish-файлів, налаштування параметрів підключення до бази даних, застосування міграцій Entity Framework Core або імпорт SQL-скрипту, запуск вебзастосунку та перевірку роботи основних функціональних модулів.

Таблиця 4.11 – Етапи розгортання системи на хостингу

№	Етап	Зміст
1	Підготовка сервера	Вибір хостингу або серверного середовища
2	Встановлення середовища	Перевірка наявності .NET Runtime
3	Підготовка БД	Створення або налаштування бази даних SQL Server

4	Завантаження файлів	Розміщення publish-пакету на сервері
5	Налаштування конфігурації	Встановлення параметрів БД і середовища
6	Застосування міграцій	Оновлення структури бази даних
7	Запуск системи	Старт вебзастосунку на сервері
8	Перевірка модулів	Авторизація, заявки, відвідуваність, аналітика та звіти

Після розгортання необхідно перевірити авторизацію користувачів, коректність рольового доступу, подання і розгляд заявок, роботу модуля відвідуваності, збереження результатів участі та формування аналітичних звітів. Окремо слід переконатися, що база даних доступна лише відповідальним особам, а доступ до вебзастосунку здійснюється через захищений канал. Супровід системи має виконуватися через вихідний код проєкту та контрольоване оновлення структури бази даних. У разі зміни моделей даних доцільно використовувати механізм міграцій Entity Framework Core. Перед впровадженням змін у робоче середовище необхідно виконувати тестування на окремій копії системи або бази даних. Подальший розвиток системи може бути спрямований на розширення аналітики, додавання нових фільтрів і показників, реалізацію експорту у PDF або Excel, розширення ролей користувачів, інтеграцію з університетськими інформаційними ресурсами, покращення інтерфейсу та створення повнішого журналу дій користувачів.

Таким чином, інсталяційний пакет повинен забезпечувати не лише технічний запуск вебзастосунку, а й повноцінне розгортання інформаційної системи разом із базою даних, конфігурацією середовища та початковими рекомендаціями для адміністрування й експлуатації. Такий підхід дозволяє підготувати систему до локального використання або подальшого хостингу в серверному середовищі.

ВИСНОВОК

У бакалаврській кваліфікаційній роботі було розроблено інформаційну систему обліку та аналізу діяльності студентських наукових гуртків. Розробка спрямована на автоматизацію процесів обліку учасників гуртків, подання та розгляду заявок, ведення відвідуваності, фіксації результатів участі студентів у наукових заходах, а також формування аналітичної звітності.

Перший розділ: проведено системний аналіз предметної області. Розглянуто особливості організації діяльності студентських наукових гуртків, визначено основні сутності, користувачів системи та інформаційні потоки. Проаналізовано існуючі рішення у сфері обліку наукової діяльності та встановлено, що вони переважно орієнтовані на індивідуальну публікаційну активність, а не на внутрішній облік діяльності студентських наукових гуртків. Також сформульовано вимоги до програмної системи та постановку завдання.

У другому розділі розроблено інформаційне забезпечення системи. Побудовано логічну модель даних, визначено основні таблиці, зв’язки між ними, первинні та зовнішні ключі. Для реалізації інформаційної бази обґрунтовано вибір Microsoft SQL Server Express, оскільки ця СУБД добре інтегрується з ASP.NET Core та Entity Framework Core, підтримує реляційну модель даних, ключі, індекси, унікальні обмеження і представлення. Описано створення бази даних StudentScienceClubsDB, використання контексту StudentScienceClubsContext та міграцій Entity Framework Core.

У третьому розділі описано прикладне програмне забезпечення. Систему реалізовано як вебзастосунок на платформі ASP.NET Core MVC. Користувацький інтерфейс побудовано на основі Razor Views, серверна логіка зосереджена в контролерах, а доступ до даних реалізовано через Entity Framework Core. Описано організаційну структуру проєкту, обраний інструментарій розробки та алгоритми основних програмних модулів:

авторизації, подання і розгляду заявок, обліку відвідуваності, формування аналітики, експорту звітів і затвердження звітності.

У четвертому розділі наведено рекомендації щодо впровадження та експлуатації системи. Проведено тестування основних модулів за тестовими сценаріями, визначено вимоги до апаратного та програмного забезпечення, а також описано склад інсталяційного пакету для розгортання вебзастосунку. Окрему увагу приділено роботі системи на хостингу, використанню publish-пакету, налаштуванню бази даних і безпечному зберіганню параметрів підключення.

У результаті виконання роботи створено програмний продукт, який забезпечує централізоване зберігання інформації про студентські наукові гуртки, учасників, заявки, засідання, відвідуваність, наукові заходи, результати участі та аналітичні звіти. Розроблена система підвищує зручність обліку діяльності гуртків, зменшує дублювання інформації, спрощує формування звітності та створює основу для подальшого розвитку, зокрема розширення форматів експорту, удосконалення аналітики та можливої інтеграції з університетськими інформаційними ресурсами.

Список використаних джерел

1. Методичні рекомендації щодо підготовки та оформлення бакалаврської кваліфікаційної роботи для студентів спеціальності 122 «Комп'ютерні науки». Київ : НУБіП України, 2026.

2. ASP.NET Core documentation. Microsoft Learn. URL: https://learn.microsoft.com/aspnet/core/

3. ASP.NET Core MVC overview. Microsoft Learn. URL: https://learn.microsoft.com/aspnet/core/mvc/overview

4. Razor syntax reference for ASP.NET Core. Microsoft Learn. URL: https://learn.microsoft.com/aspnet/core/mvc/views/razor (дата звернення: 19.05.2026).

5. Entity Framework Core documentation. Microsoft Learn. URL: https://learn.microsoft.com/ef/core/

6. Entity Framework Core migrations overview. Microsoft Learn. URL: https://learn.microsoft.com/ef/core/managing-schemas/migrations/

7. SQL Server documentation. Microsoft Learn. URL: https://learn.microsoft.com/sql/sql-server/

8. SQL Server Express LocalDB. Microsoft Learn. URL: https://learn.microsoft.com/sql/database-engine/configure-windows/sql-server-express-localdb

9. C# documentation. Microsoft Learn. URL: https://learn.microsoft.com/dotnet/csharp/

10. .NET documentation. Microsoft Learn. URL: https://learn.microsoft.com/dotnet/

11. Authentication and authorization in ASP.NET Core. Microsoft Learn. URL: https://learn.microsoft.com/aspnet/core/security/authentication/

12. OWASP Web Security Testing Guide. OWASP Foundation. URL: https://owasp.org/www-project-web-security-testing-guide/

13. Unified Modeling Language specification. Object Management Group. URL: https://www.omg.org/spec/UML/

14. PlantUML documentation. URL: https://plantuml.com/

15. ORCID. URL: https://orcid.org/

16. Google Scholar. URL: https://scholar.google.com/

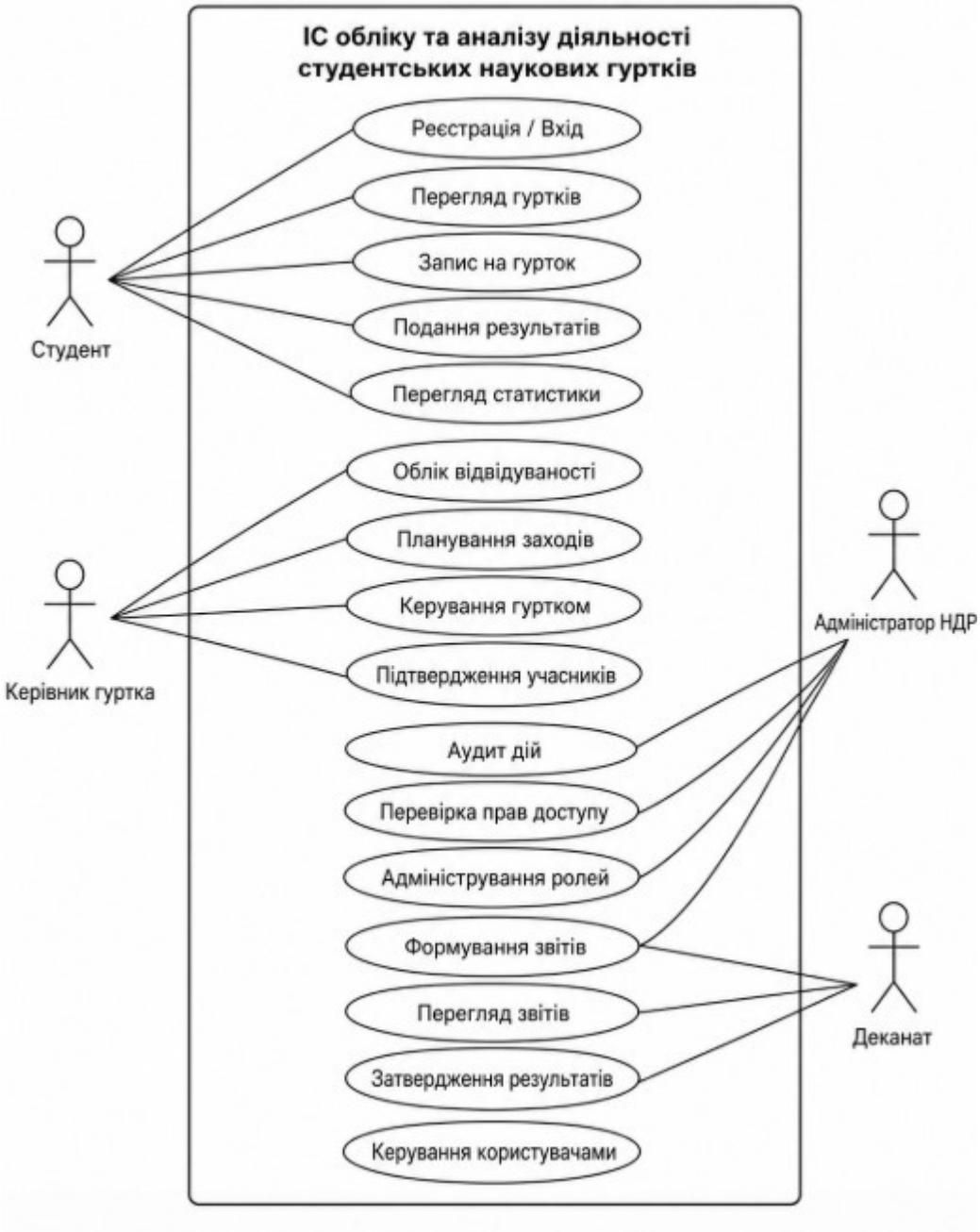
17. Scopus preview. Elsevier. URL: https://www.scopus.com/

18. Web of Science. Clarivate. URL: https://www.webofscience.com/

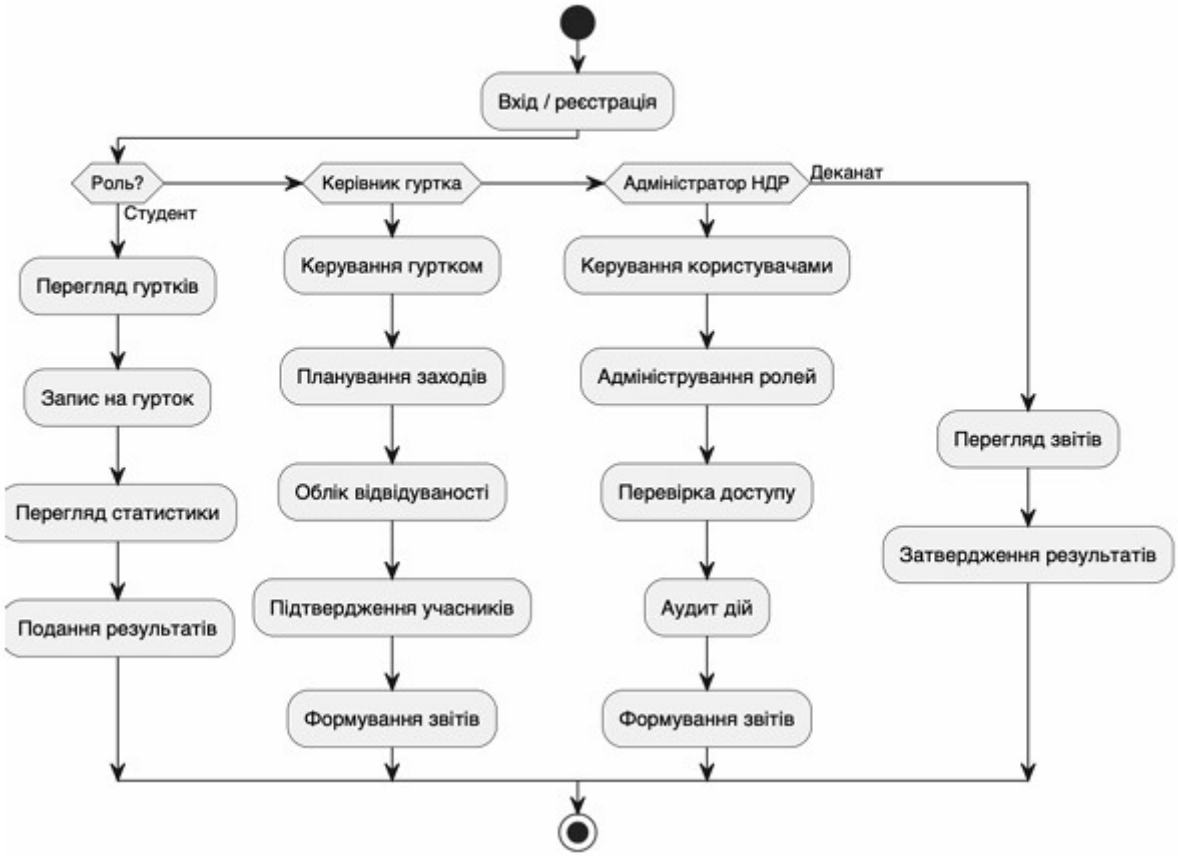
19. Visual Studio documentation. Microsoft Learn. URL: https://learn.microsoft.com/visualstudio/

20. Git documentation. URL: https://git-scm.com/doc

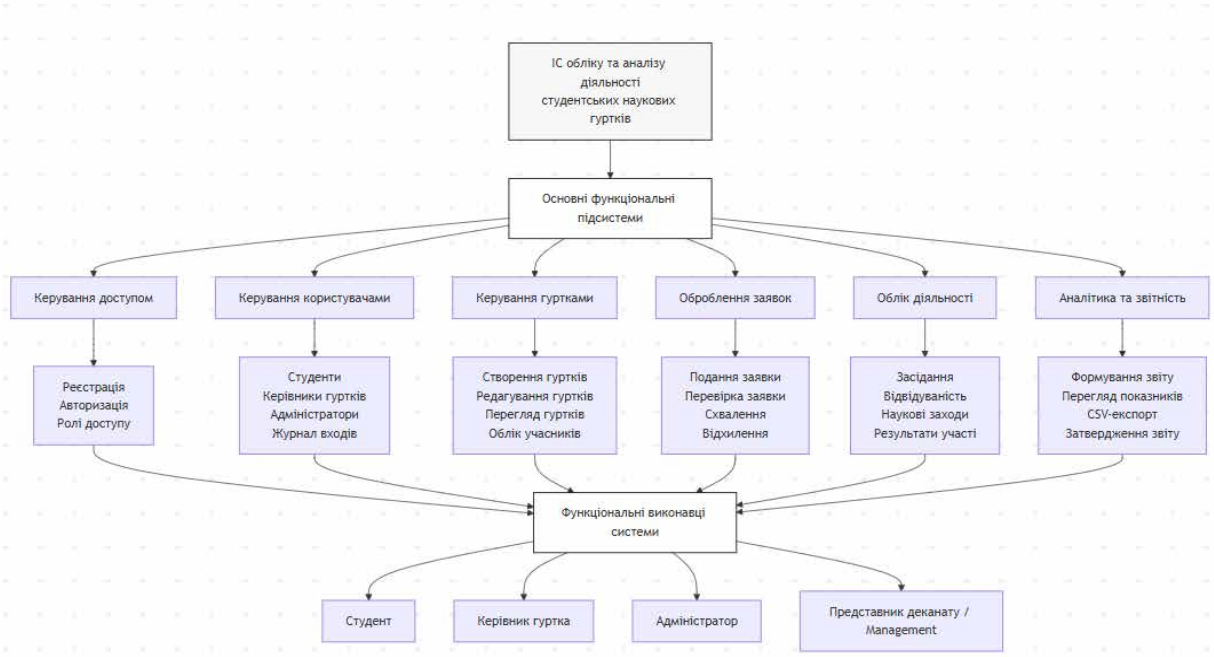
Додаток А – Діаграма прецедентів



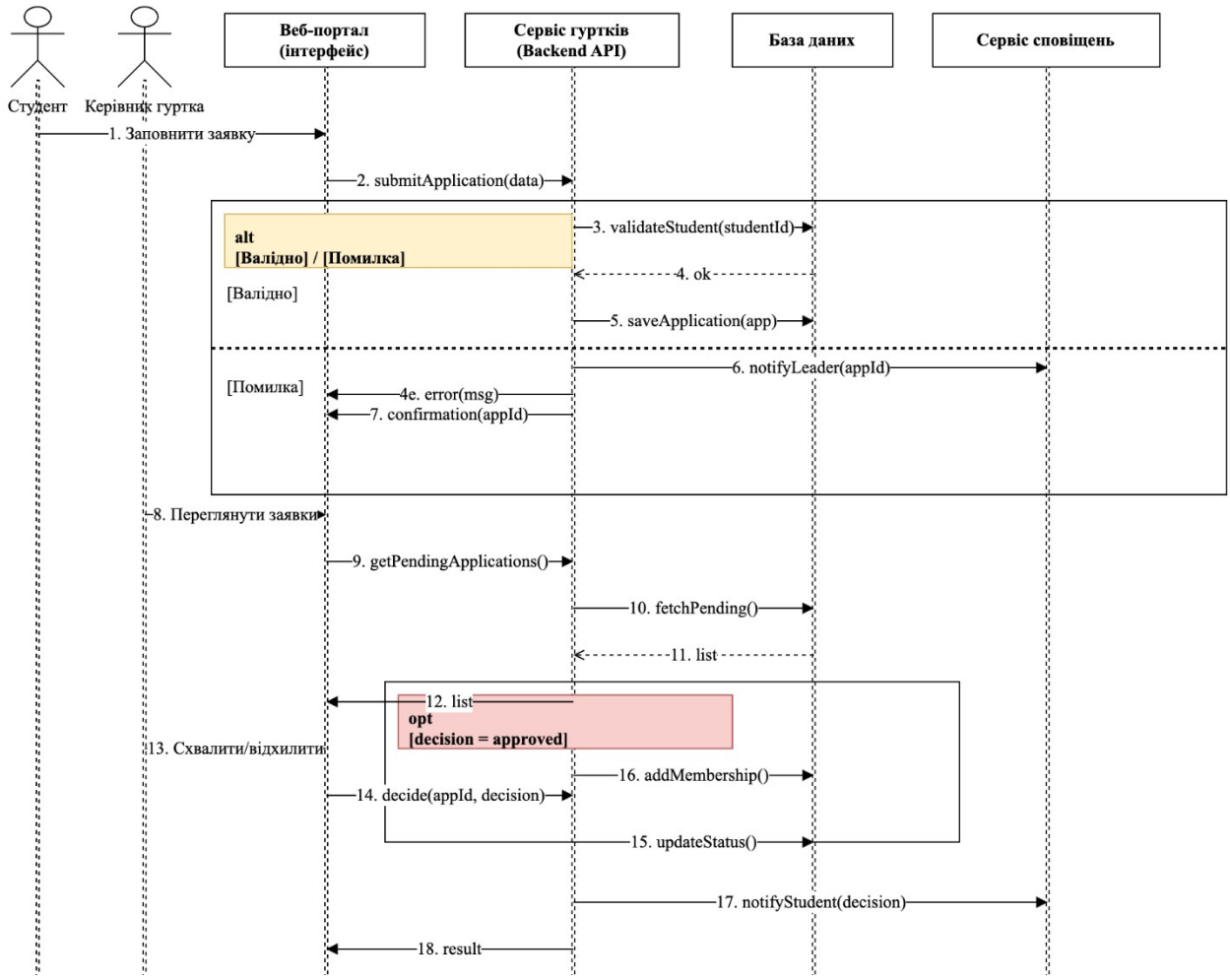
Додаток Б – Діаграма діяльності



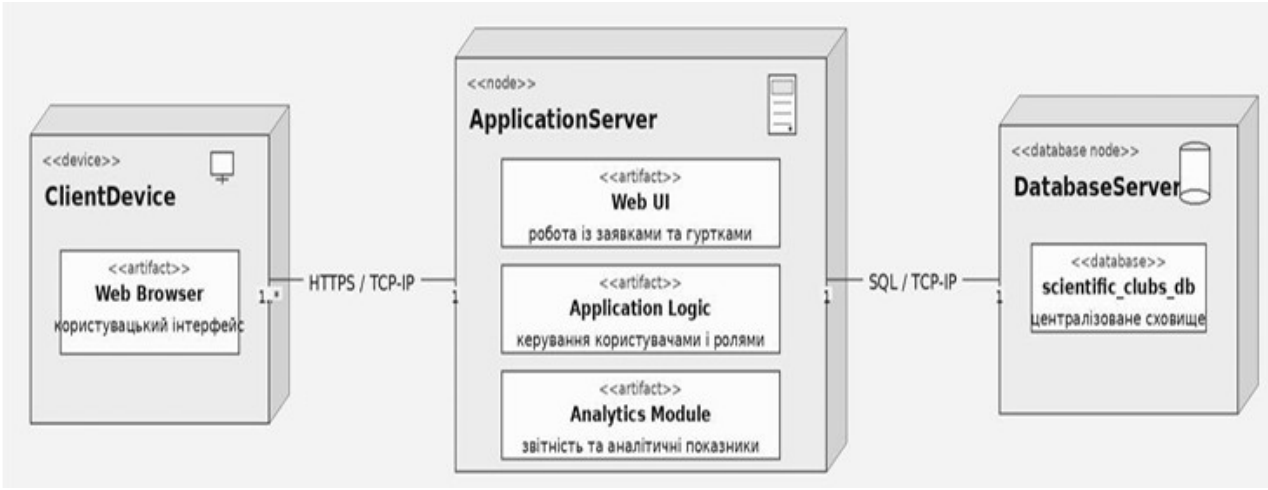
Додаток В – Функціональна діаграма



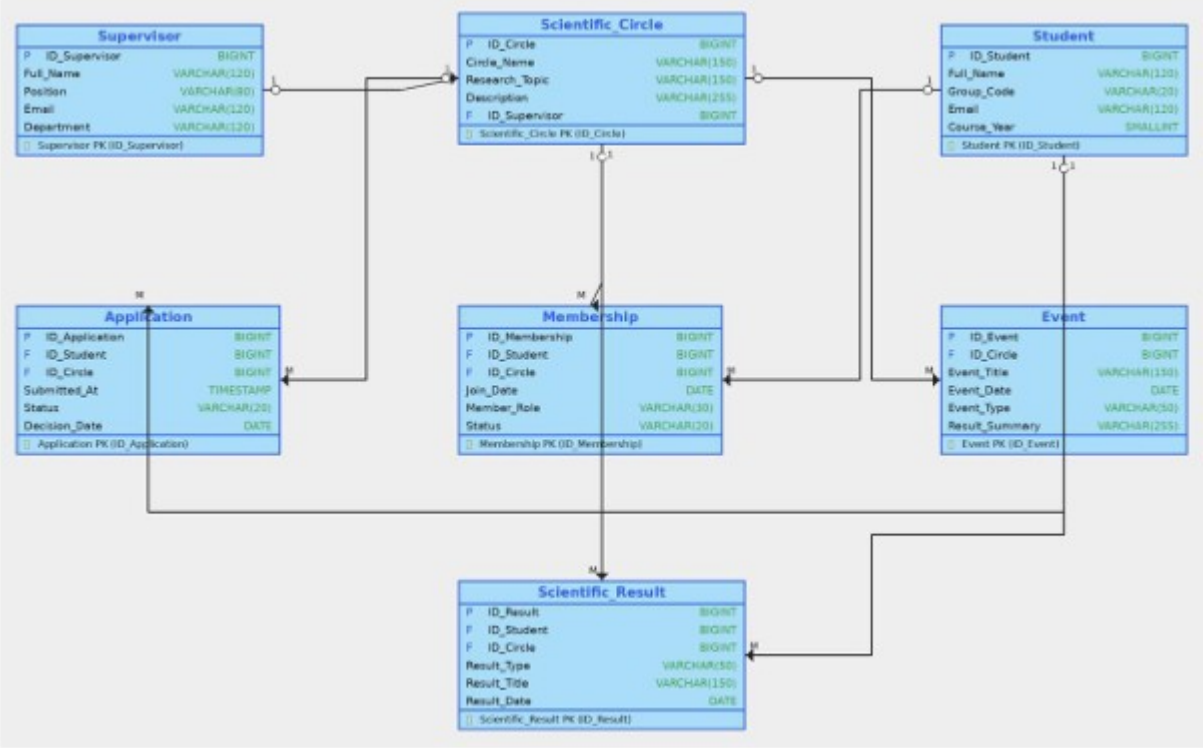
Додаток Г – Діаграма послідовності



Додаток Д – Діаграма розгортання



Додаток Е – Логічна модель БД



Додаток Ж – Program.cs

```
using Microsoft.AspNetCore.Authentication.Cookies;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc.Authorization;
using Microsoft.EntityFrameworkCore;
using Диплом_1.Data;
using Диплом_1.Models;

using System.Data;
using System.Data.Common;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme)
    .AddCookie(options =>
    {
        options.LoginPath = "/Account/Login";
        options.AccessDeniedPath = "/Account/AccessDenied";
        options.ExpireTimeSpan = TimeSpan.FromHours(8);
    });

builder.Services.AddControllersWithViews(options =>
{
    var policy = new AuthorizationPolicyBuilder()
        .RequireAuthenticatedUser()
        .Build();

    options.Filters.Add(new AuthorizeFilter(policy));
});

builder.Services.AddDbContext<StudentScienceClubsContext>(options =>
    options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));

var app = builder.Build();

await SeedAuthDataAsync(app);
```

```
// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Home/Error");
    // The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.
    app.UseHsts();
}

app.UseHttpsRedirection();
app.UseRouting();

app.UseAuthentication();
app.UseAuthorization();

app.MapStaticAssets();

app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}")
    .WithStaticAssets();


app.Run();

static async Task SeedAuthDataAsync(WebApplication app)
{
    using var scope = app.Services.CreateScope();
    var context = scope.ServiceProvider.GetRequiredService<StudentScienceClubsContext>();

    await EnsureDatabaseSchemaAsync(context);

    var firstStudent = await context.Students
        .OrderBy(student => student.StudentId)
        .FirstOrDefaultAsync();

    var firstSupervisor = await context.Supervisors
        .OrderBy(supervisor => supervisor.SupervisorId)
        .FirstOrDefaultAsync();
```

```
            await SeedUserAsync(context, "admin", "admin123", "Admin",
"Адміністратор", null, null);
        await SeedUserAsync(
            context,
            "student",
            "student123",
            "Student",
                firstStudent == null ? "Студент" : $"{firstStudent.LastName}
{firstStudent.FirstName}",
            firstStudent?.StudentId,
            null);
        await SeedUserAsync(
            context,
            "supervisor",
            "supervisor123",
            "Supervisor",
                            firstSupervisor == null ? "Керівник гуртка" :
$"{firstSupervisor.LastName} {firstSupervisor.FirstName}",
            null,
            firstSupervisor?.SupervisorId);
                await SeedUserAsync(context, "management", "management123",
"Management", "Деканат", null, null);
        await SeedStudentAccountsAsync(context);
        await SeedSupervisorAccountsAsync(context);
        await EnsureEveryClubHasSupervisorAsync(context);
    }

    static async Task EnsureDatabaseSchemaAsync(StudentScienceClubsContext
context)
    {
        await MarkExistingDatabaseAsMigratedAsync(context);
        await context.Database.MigrateAsync();
    }

    static                                async                                Task
MarkExistingDatabaseAsMigratedAsync(StudentScienceClubsContext context)
    {
        const string initialMigrationId = "20260516201252_InitialSchema";
        const string productVersion = "10.0.6";

        var connection = context.Database.GetDbConnection();
        var shouldCloseConnection = connection.State != ConnectionState.Open;
```

```
        if (shouldCloseConnection)
        {
            await connection.OpenAsync();
        }

        try
        {
                var hasMigrationHistory = await TableExistsAsync(connection,
"__EFMigrationsHistory");
            if (hasMigrationHistory)
            {
                return;
            }

            var hasExistingSchema =
                await TableExistsAsync(connection, "Faculties") ||
                await TableExistsAsync(connection, "Departments") ||
                await TableExistsAsync(connection, "Students") ||
                await TableExistsAsync(connection, "ScienceClubs");

            if (!hasExistingSchema)
            {
                return;
            }

            await using var command = connection.CreateCommand();
            command.CommandText =
                """
                CREATE TABLE [__EFMigrationsHistory]
                (
                    [MigrationId] nvarchar(150) NOT NULL,
                    [ProductVersion] nvarchar(32) NOT NULL,
                        CONSTRAINT [PK___EFMigrationsHistory] PRIMARY KEY
([MigrationId])
                );

                        INSERT INTO [__EFMigrationsHistory] ([MigrationId],
[ProductVersion])
                VALUES (@migrationId, @productVersion);
                """;

            var migrationIdParameter = command.CreateParameter();
            migrationIdParameter.ParameterName = "@migrationId";
```

```
            migrationIdParameter.Value = initialMigrationId;
            command.Parameters.Add(migrationIdParameter);

            var productVersionParameter = command.CreateParameter();
            productVersionParameter.ParameterName = "@productVersion";
            productVersionParameter.Value = productVersion;
            command.Parameters.Add(productVersionParameter);

            await command.ExecuteNonQueryAsync();
        }
        finally
        {
            if (shouldCloseConnection)
            {
                await connection.CloseAsync();
            }
        }
    }

    static async Task<bool> TableExistsAsync(DbConnection connection, string tableName)
    {
        await using var command = connection.CreateCommand();
        command.CommandText = "SELECT COUNT(1) FROM sys.tables WHERE [name] = @tableName";

        var tableNameParameter = command.CreateParameter();
        tableNameParameter.ParameterName = "@tableName";
        tableNameParameter.Value = tableName;
        command.Parameters.Add(tableNameParameter);

        var result = await command.ExecuteScalarAsync();
        return Convert.ToInt32(result) > 0;
    }

    static async Task SeedUserAsync(
        StudentScienceClubsContext context,
        string userName,
        string password,
        string roleName,
        string displayName,
        int? studentId,
        int? supervisorId)
```

```
        {
            var existingUser = await context.AppUsers.FirstOrDefaultAsync(user => user.UserName == userName);
            if (existingUser != null)
            {
                if (existingUser.StudentId == null && studentId.HasValue)
                {
                    existingUser.StudentId = studentId;
                    existingUser.DisplayName = displayName;
                }

                if (existingUser.SupervisorId == null && supervisorId.HasValue)
                {
                    existingUser.SupervisorId = supervisorId;
                    existingUser.DisplayName = displayName;
                }

                await context.SaveChangesAsync();
                return;
            }

            var user = new AppUser
            {
                UserName = userName,
                RoleName = roleName,
                DisplayName = displayName,
                StudentId = studentId,
                SupervisorId = supervisorId,
                IsActive = true,
                CreatedAt = DateTime.Now
            };

                                                    user.PasswordHash = new PasswordHasher<AppUser>().HashPassword(user, password);

            context.AppUsers.Add(user);
            await context.SaveChangesAsync();
        }

        static async Task SeedStudentAccountsAsync(StudentScienceClubsContext context)
        {
            var students = await context.Students
```

```
            .Where(student => student.IsActive)
            .OrderBy(student => student.StudentId)
            .ToListAsync();

        var passwordHasher = new PasswordHasher<AppUser>();

        foreach (var student in students)
        {
                var alreadyLinked = await context.AppUsers.AnyAsync(user =>
user.StudentId == student.StudentId);
            if (alreadyLinked)
            {
                continue;
            }

            var userName = $"student{student.StudentId}";
            var suffix = 1;

                while (await context.AppUsers.AnyAsync(user => user.UserName ==
userName))
            {
                userName = $"student{student.StudentId}_{suffix}";
                suffix++;
            }

            var displayName = $"{student.LastName} {student.FirstName}";
            var user = new AppUser
            {
                UserName = userName,
                RoleName = "Student",
                DisplayName = displayName,
                StudentId = student.StudentId,
                SupervisorId = null,
                IsActive = true,
                CreatedAt = DateTime.Now
            };

                        user.PasswordHash = passwordHasher.HashPassword(user,
"student123");
            context.AppUsers.Add(user);
        }

        await context.SaveChangesAsync();
```

```
    }

    static async Task SeedSupervisorAccountsAsync(StudentScienceClubsContext
context)
    {
        var supervisors = await context.Supervisors
            .OrderBy(supervisor => supervisor.SupervisorId)
            .ToListAsync();

        var passwordHasher = new PasswordHasher<AppUser>();

        foreach (var supervisor in supervisors)
        {
                var alreadyLinked = await context.AppUsers.AnyAsync(user =>
user.SupervisorId == supervisor.SupervisorId);
            if (alreadyLinked)
            {
                continue;
            }

            var userName = $"supervisor{supervisor.SupervisorId}";
            var suffix = 1;

                while (await context.AppUsers.AnyAsync(user => user.UserName ==
userName))
            {
                userName = $"supervisor{supervisor.SupervisorId}_{suffix}";
                suffix++;
            }

            var displayName = $"{supervisor.LastName} {supervisor.FirstName}";
            var user = new AppUser
            {
                UserName = userName,
                RoleName = "Supervisor",
                DisplayName = displayName,
                StudentId = null,
                SupervisorId = supervisor.SupervisorId,
                IsActive = true,
                CreatedAt = DateTime.Now
            };
```

```
                user.PasswordHash = passwordHasher.HashPassword(user,
"supervisor123");
            context.AppUsers.Add(user);
        }

        await context.SaveChangesAsync();
    }

    static async Task
EnsureEveryClubHasSupervisorAsync(StudentScienceClubsContext context)
    {
        var supervisorIds = await context.Supervisors
            .Select(supervisor => supervisor.SupervisorId)
            .ToListAsync();

        if (supervisorIds.Count == 0)
        {
            return;
        }

        var fallbackSupervisorId = supervisorIds[0];
        var validSupervisorIds = supervisorIds.ToHashSet();

        var clubsToFix = await context.ScienceClubs
            .Where(club => !validSupervisorIds.Contains(club.SupervisorId))
            .ToListAsync();

        if (clubsToFix.Count == 0)
        {
            return;
        }

        foreach (var club in clubsToFix)
        {
            club.SupervisorId = fallbackSupervisorId;
        }

        await context.SaveChangesAsync();
    }
```

Додаток З – Декларація ШІ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

Декларація про академічну доброчесність та використання ШІ

Я, _____Липницький Ярослав Андрійович_____, здобувач НУБіП України, усвідомлюючи відповідальність за порушення академічної доброчесності, цією заявою підтверджую, що: 1. Моя бакалаврська кваліфікаційна робота на тему «Інформаційна система обліку та аналізу діяльності студентських наукових гуртків» **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.

__Ярослав Липницький_____ Ім'я ПРІЗВИЩЕ