Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

КОМП'ЮТЕРНА СИСТЕМА КОНТРОЛЮ ДОСТУПУ ДО СПОРТИВНОГО КОМПЛЕКСУ З ВИКОРИСТАННЯМ МЕРЕЖЕВИХ СЕРВІСІВ

Виконав: студент 2 курсу, групи 2КІ-24м
спеціальності 123 — «Комп'ютерна інженерія»
_____ Доманський А. Ф.

Керівник: к.т.н.., доц. каф. ОТ
_____ Кожем'яко А. В.
«15» 12 2025 р.

Опонент: к.т.н., доц. каф. ПЗ
_____ Коваленко О. О.
«15» 12 2025 р.

Допущено до захисту

Завідувач кафедри ОТ
д.т.н., проф. Азаров О. Д.
«18» 12 2025 р.

ВНТУ 2025

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Галузь знань — Інформаційні технології
Освітній рівень — магістр
Спеціальність — 123 Комп'ютерна інженерія
Освітньо-професійна програма — Комп'ютерна інженерія

ЗАТВЕРДЖУЮ

Завідувач кафедри обчислювальної техніки

_____ О.Д. Азаров

" 25 " вересня 2025 р.

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Доманському Антону Францовичу

1 Тема роботи «Комп'ютерна система контролю доступу до спортивного комплексу з використанням мережевих сервісів» керівник роботи Кожем'яко А. В. к.т.н. доцент кафедри ОТ, затверджено наказом вищого навчального закладу від 24.09.25 року №313.

2 Строк подання студентом роботи 18.12.25.

3 Вихідні дані до роботи: призначення: забезпечення контролю доступу та моніторингу активності користувачів у спортивному комплексі з використанням веб-технологій та мережевих сервісів., засоби — середовище програмування Visual Studio Code, мови програмування JavaScript та TypeScript, фреймворк React, Node.js, база даних SQLite, бібліотеки Chart.js.

4 Зміст розрахунково-пояснювальної: аналіз сучасних методів та технологій контролю доступу у спортивних комплексах, аналіз та вибір інструментів для розробки веб-додатку адміністратора, розробка структури бази даних користувачів, створення веб-інтерфейсу адміністратора та панелі моніторингу, тестування та аналіз роботи веб-додатку.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): блок-схема апаратної частини СКУД, ER-діаграма бази даних, структурна схема розподіленої комп'ютерної системи, UML-модель процесу позиціонування, блок-схема алгоритму клієнтської частини

6 Консультанти розділів роботи приведені в таблиці 1.

Таблиця 1— Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Кожем'яко Андрій Вікторович к.т.н., доцент	25.09.2025	17.12.2025
5	Адлер Оксана Олександрівна к.т.н., доцент	25.09.2025	17.12.2025
Нормоконтроль	асист. каф. ОТ Швець С. І.		

7 Дата видачі завдання **25.09.2025**.

8 Календарний план виконання МКР приведений в таблиці 2.

Таблиця 2 — Календарний план

№ з/п	Назва етапів МКР	Строк виконання	Підпис
1	Постановка задачі	26.09.25	
2	Огляд існуючих рішень	29.09 - 06.10	
3	Аналіз аналогової частини	07.10 - 15.10	
4	Вибір ПЗ для розробки	16.10 - 17.10	
5	Розробка комп'ютерної системи	20.10 - 29.10	
6	Розрахунок економічної частини	03.11 - 11.11	
7	Оформлення пояснювальної записки та ілюстративного матеріалу	12.11 - 20.11	
8	Виконання магістерської кваліфікаційної роботи	24.11.25	
9	Перевірка якості виконання магістерської кваліфікаційної роботи та усунення недоліків	01.12.25	
10	Підписи супроводжувальних документів у керівника, опонента, нормоконтролера	15.12.25	
11	Перевірка «антиплагіат»	15.12.25	
12	Попередній захист	12.11.25	

Студент Доманський Антон Францович

Керівник к.т.н., доц. Кожем'яко Андрій Вікторович

АНОТАЦІЯ

УДК 004

Доманський А. Ф. Комп’ютерна система контролю доступу до спортивного комплексу з використанням мережевих сервісів. Магістерська кваліфікаційна робота зі спеціальності 123 — Комп’ютерна Інженерія, Вінниця: ВНТУ, 2025 — 122 с. На укр. мові. Бібліогр.: 31 назв; рис.: 21; табл. 10.

У роботі розглянуто сучасні методи та технології контролю доступу та моніторингу активності користувачів у спортивних комплексах, обґрунтовано вибір технологій та середовищ розробки веб-додатку адміністратора, реалізовано серверну та клієнтську частини системи. Створено інтерактивний веб-інтерфейс для адміністратора, який дозволяє переглядати карту об’єкта, статистику користувачів, журнали подій та керувати правами доступу в реальному часі. Розроблено механізми додавання та редагування користувачів, контролю їхніх прав доступу та фіксації системних подій. Проведено тестування веб-додатку та надано рекомендації щодо подальшого вдосконалення системи.

Ключові слова: контроль доступу, користувачі, веб-додаток, система адміністрування, локальна мережа, моніторинг активності.

ABSTRACT

Domanskyi A. F. Computer Access Control System for a Sports Complex Using Network Services. Master's Thesis in Specialty 123 — Computer Engineering, Vinnytsia: VNTU, 2025 — 122 p. In Ukrainian. Bibliography: 31 titles; figures: 21; tables: 10.

The thesis examines modern methods and technologies for access control and user activity monitoring in sports complexes, justifies the choice of technologies and development environments for the administrator's web application, and implements the server-side and client-side components of the system. An interactive web interface for the administrator has been created, allowing real-time viewing of the facility map, user statistics, event logs, and management of access rights. Mechanisms for adding and editing users, controlling their access rights, and recording system events have been developed. Testing of the web application has been conducted, and recommendations for further system improvement are provided.

Keywords: access control, users, web application, administration system, local network, activity monitoring.

ЗМІСТ

ВСТУП

У сучасних умовах цифровізації та розвитку інформаційних технологій питання забезпечення безпеки та автоматизації процесів управління доступом до приміщень набуває особливої актуальності. Спортивні комплекси, як об’єкти з великим потоком відвідувачів, потребують ефективних рішень для контролю доступу, обліку користувачів і моніторингу подій у режимі реального часу. Традиційні методи контролю, засновані на використанні ключів або механічних перепусток, мають низку недоліків — від складності адміністрування до низького рівня безпеки.

Застосування комп’ютерних систем контролю доступу, інтегрованих із сучасними мережевими сервісами, дозволяє значно підвищити рівень безпеки, зручність користування та ефективність управління спортивними об’єктами. Такі системи забезпечують ідентифікацію користувачів за допомогою електронних карток, RFID-міток чи біометричних даних, а також дозволяють здійснювати централізований контроль і аналітику через веб-інтерфейс або мобільний застосунок.

Розробка комп’ютерної системи контролю доступу до спортивного комплексу є актуальним завданням, що поєднує знання в галузі комп’ютерних мереж, програмування, інформаційної безпеки та системного адміністрування. Такий підхід забезпечує не лише контроль доступу до приміщень, а й інтеграцію з іншими сервісами — наприклад, системами обліку відвідувань, бронювання тренувань або ведення статистики.

Актуальність теми дослідження полягає в тому, що завдяки розвитку сучасних інформаційних та мережевих технологій з’явилася можливість ефективно автоматизувати процес контролю доступу до різноманітних об’єктів, зокрема спортивних комплексів. Використання комп’ютерних систем у поєднанні з мережевими сервісами дозволяє створити гнучке, безпечне та зручне середовище для керування відвідуваннями, забезпечуючи ідентифікацію користувачів, моніторинг у реальному часі та збереження історії подій.

Такі системи значно спрощують роботу адміністрації спортивного комплексу, забезпечують оперативний облік клієнтів, дозволяють контролювати навантаження на зали та обмежувати доступ до окремих зон залежно від рівня користувача або розкладу

занять. Крім того, інтеграція мережевих сервісів надає можливість віддаленого керування системою, отримання звітів через веб-інтерфейс, а також підвищує рівень інформаційної безпеки завдяки централізованому зберіганню даних.

Метою роботи є розширення функціональних можливостей комп'ютерної системи контролю доступу до спортивного комплексу з використанням мережевих сервісів, за рахунок інтеграції локальних пристроїв контролю з центральним сервером через веб-сервіси, що забезпечить автоматизований облік користувачів, підвищений рівень безпеки та ефективне управління доступом до об'єктів інфраструктури.

Для досягнення поставленої мети у роботі розв'язуються такі задачі:

— аналіз сучасних систем контролю доступу та технології, що використовуються у спортивних закладах;

— дослідження використання мережевих сервісів для забезпечення взаємодії між компонентами системи;

— розробка архітектури комп'ютерної системи контролю доступу з урахуванням вимог безпеки та масштабованості;

— розробка програмного забезпечення системи з використанням сучасних інструментів веб-розробки та мережевих технологій.

Об'єктом дослідження є процес контролю доступу до об'єктів спортивного комплексу із застосуванням комп'ютерних та мережевих технологій.

Предметом дослідження є програмні засоби та мережеві сервіси, що забезпечують реалізацію комп'ютерної системи контролю доступу.

Методи дослідження: методи системного аналізу — для дослідження структури та взаємодії компонентів системи; методи алгоритмічного проєктування — для розробки логіки обробки запитів доступу; методи мережевого моделювання — для забезпечення коректної взаємодії між клієнтськими пристроями та сервером; методи експериментального тестування — для перевірки працездатності системи в умовах реального функціонування спортивного комплексу.

Наукова новизна роботи полягає в тому, що набув подальшого розвитку метод адаптивного вирішення конфліктів під час синхронізації даних між офлайн-контролерами спортивного комплексу та центральним сервером. Запропонований метод, на відміну від існуючих підходів, передбачає динамічний аналіз часових відміток, пріоритетів подій та ступеня достовірності локальних журналів, що дозволяє мінімізувати втрати інформації та уникати дублювання записів у разі нестабільного або переривчастого мережевого з'єднання.

Практичне значення роботи полягає в тому, що впровадження розробленої комп'ютерної системи контролю доступу до спортивного комплексу дозволяє автоматизувати процес ідентифікації та реєстрації відвідувачів, підвищити рівень безпеки об'єкта та зменшити кількість випадків несанкціонованого проникнення.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ КОНТРОЛЮ ДОСТУПУ

1.1 Специфіка контролю доступу в спортивних комплексах

Сучасні спортивні комплекси є складними інженерно-інформаційними об’єктами, які об’єднують в собі велику кількість функціональних зон: тренажерні зали, басейни, майданчики для командних видів спорту, зали єдиноборств, фітнес-зони, роздягальні, душові, адміністративні приміщення, а також технічні кімнати. Для ефективного управління такими об’єктами необхідно застосовувати автоматизовані системи контролю доступу (СКД), які дозволяють забезпечити безпечний, впорядкований і зручний доступ для різних категорій користувачів [1].

Особливість контролю доступу у спортивних комплексах полягає в тому, що одночасно в приміщенні можуть перебувати сотні людей — клієнтів, тренерів, персоналу та відвідувачів заходів. Тому система повинна забезпечувати швидке розпізнавання користувачів та оперативну обробку запитів на вхід і вихід, не створюючи черг у пікові години. Для цього застосовуються турнікети зі швидкодіючими зчитувачами, які підтримують роботу з RFID-картами, браслетами, мобільними додатками або QR-кодами [1].

Другою важливою специфікою є необхідність гнучкого розмежування рівнів доступу. Наприклад, клієнти можуть мати право входу лише до зони тренувань чи басейну, тоді як обслуговуючий персонал повинен мати можливість входити в технічні приміщення. Адміністрація, у свою чергу, потребує повного доступу до системи для моніторингу, звітності та управління клієнтськими записами. Тому сучасні СКД повинні підтримувати створення ролей і профілів користувачів, що відповідають їхнім повноваженням [1].

Ще однією важливою особливістю є інтеграція системи контролю доступу з іншими інформаційними підсистемами спортивного комплексу. Це можуть бути модулі ведення клієнтської бази, обліку відвідувань, управління абонементами, системи розрахунків та аналітики. Завдяки цьому створюється єдина цифрова екосистема, що дозволяє автоматизувати не лише контроль відвідування, а й фінансові процеси, а також покращити рівень сервісу для користувачів.

Важливо також забезпечити безконтактний принцип доступу (рисунок 1.1). Використання безконтактних карток, RFID-міток, біометричних ідентифікаторів або мобільних додатків дозволяє зменшити час на проходження через турнікети та підвищити санітарно-епідеміологічну безпеку, що особливо актуально після пандемії COVID-19.

Рисунок 1.1 — Біометричний зчитувач

Система контролю доступу у спортивних комплексах повинна враховувати специфіку роботи з великою кількістю постійних і тимчасових клієнтів. Наприклад, відвідувач може придбати одноразовий вхід або абонемент на певну кількість занять, а система має автоматично фіксувати кожен прохід і блокувати доступ після вичерпання ліміту. Це дозволяє уникнути зловживань і підвищує прозорість взаємодії між клієнтами та адміністрацією [2].

Ще одним суттєвим аспектом є забезпечення безпеки майна та відвідувачів. Система контролю доступу має фіксувати всі події у журналі, зберігати історію входів і виходів, а також забезпечувати інтеграцію з відеоспостереженням, пожежною сигналізацією та охоронними службами. Такий підхід дає змогу швидко реагувати на позаштатні ситуації й підвищує рівень загальної безпеки об’єкта.

З технічної точки зору, специфіка контролю доступу у спортивних комплексах також передбачає необхідність стабільної роботи мережевої інфраструктури. Системи повинні мати можливість централізованого керування через локальні або хмарні сервіси, що забезпечує зручність адміністрування, моніторингу та оновлення програмного забезпечення [2].

Отже, специфіка контролю доступу у спортивних комплексах полягає в поєднанні вимог до безпеки, зручності, масштабованості та інтегрованості системи. Такі комплекси потребують сучасних мережевих рішень, здатних ефективно керувати потоками користувачів, забезпечувати надійний облік відвідувань та підтримувати високу швидкість і точність ідентифікації при мінімальних витратах людських ресурсів.

1.2 Огляд існуючих комп’ютерних систем контролю доступу

Системи контролю та управління доступом (СКУД) є невід’ємною складовою сучасної інфраструктури безпеки на об’єктах різного призначення. Їх основне завдання полягає у забезпеченні контролю за переміщенням людей або транспортних засобів, розмежуванні прав доступу, а також у веденні обліку відвідувань. У спортивних комплексах ці системи відіграють ключову роль у впорядкуванні потоків клієнтів, підвищенні рівня безпеки та автоматизації процесів обслуговування [2].

Існуючі СКУД значно відрізняються за функціональністю, архітектурою, принципами побудови та рівнем інтеграції з іншими інформаційними системами. Найпростіші з них реалізують лише базові функції перевірки прав доступу, тоді як сучасні комплексні рішення підтримують роботу з мережевими сервісами, хмарними базами даних, мобільними додатками та аналітичними модулями [2].

Класичні системи контролю доступу складаються з апаратної частини — зчитувачів, контролерів, турнікетів або електрозамків, — та програмного забезпечення, яке виконує функції адміністрування, моніторингу та зберігання даних (рисунок 1.1). Взаємодія цих елементів забезпечує безперервний цикл контролю: від ідентифікації користувача до реєстрації факту проходу в базі даних.

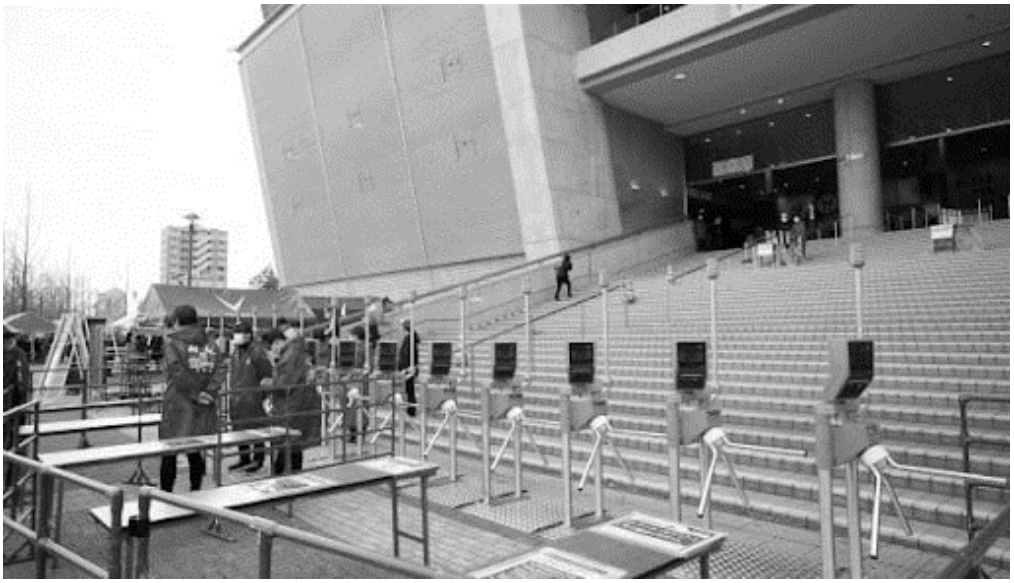

Рисунок 1.2 — Приклад апаратної частини СКУД

На сучасному етапі розвитку технологій прості автономні системи поступово витісняються мережевими рішеннями, які дозволяють централізовано керувати кількома точками доступу. Це особливо важливо для спортивних комплексів, де одночасно функціонує багато зон — тренажерні зали, басейни, фітнес-зали, адміністративні приміщення.

Одним із основних елементів будь-якої СКУД є зчитувач ідентифікаторів (рисунок 1.1). Сучасні зчитувачі використовують безконтактні технології RFID або NFC, що дозволяють швидко розпізнавати користувачів за допомогою карток, браслетів чи мобільних пристроїв. Завдяки цьому забезпечується висока швидкість проходу, що є важливим у години пікового навантаження [2].

У багатьох спортивних центрах впроваджуються біометричні системи доступу, які розпізнають користувача за відбитком пальця або обличчям. Такі рішення гарантують підвищений рівень безпеки, адже біометричні параметри не можна передати іншій особі. Разом з тим, вони потребують точного калібрування та захисту персональних даних відповідно до чинного законодавства [3].

Рисунок 1.3 — Приклад зчитувача ідентифікаторів

Ключову роль у СКУД відіграють контролери, які обробляють сигнали від зчитувачів і приймають рішення про дозвіл чи заборону проходу. Вони можуть бути як автономними, так і мережевими. У другому випадку контролери під’єднуються до центрального сервера через Ethernet або Wi-Fi, що дозволяє оперативно змінювати налаштування і синхронізувати події [3].

Для фізичного контролю проходу найчастіше використовуються турнікети. Вони бувають триподи, роторні, хвірткові або шлюзові. Турнікети не лише виконують функцію обмеження доступу, а й фіксують кількість відвідувачів, дозволяючи вести статистику використання приміщень. Важливо, щоб турнікети мали високу пропускну здатність та підтримували інтеграцію з основною СКУД [3].

Окрім турнікетів, у спортивних закладах часто використовуються електронні замки, якими керують через контролери. Вони встановлюються на дверях до роздягалень, технічних приміщень або тренерських кабінетів. Такі замки забезпечують обмежений доступ для певних категорій персоналу.

Не менш важливим компонентом є програмне забезпечення СКУД. Воно дозволяє адміністратору створювати базу користувачів, надавати або відкликати права доступу, формувати звіти, аналізувати історію відвідувань. Програмні рішення бувають як локальними (встановлюються на сервер комплексу), так і хмарними (працюють

через Інтернет). В останні роки популярності набули веб-орієнтовані системи контролю доступу, що забезпечують можливість віддаленого управління з будь-якого пристрою через браузер. Це зручно для великих мереж спортивних клубів, які можуть централізовано контролювати всі філії [3].

Особливу роль відіграють мережеві сервіси та API-інтерфейси, які забезпечують інтеграцію СКУД із зовнішніми інформаційними системами — CRM, платіжними платформами, системами бронювання або відеоспостереження. Це дозволяє створювати єдиний інформаційний простір, де всі процеси взаємопов’язані. Сучасні розробники СКУД орієнтуються на впровадження мобільних додатків, за допомогою яких користувач може входити до комплексу, переглядати свій абонемент, продовжувати його або сплачувати за послуги онлайн. Такий підхід підвищує рівень автоматизації й зменшує потребу у касирах чи адміністраторах [4].

Одним із перспективних напрямів є використання хмарних технологій. Вони забезпечують централізоване зберігання даних, автоматичне резервне копіювання та доступ до системи з будь-якої точки світу. Хмарні СКУД мають нижчу вартість впровадження, оскільки не потребують потужного локального обладнання. Багато сучасних систем підтримують аналітичні модулі, що дозволяють визначати пікові години відвідувань, аналізувати динаміку клієнтського потоку, а також прогнозувати завантаженість залів. Такі можливості допомагають адміністрації приймати обґрунтовані управлінські рішення [4].

Однією з найвідоміших комерційних систем є PERCo-Web — це мережева СКУД, яка забезпечує контроль доступу, облік робочого часу та інтеграцію з відеоспостереженням. Іншим популярним рішенням є ZKTeco, яке пропонує широкий асортимент біометричних терміналів і хмарне керування доступом [5].

Для великих спортивних об’єктів застосовуються рішення класу enterprise, такі як Hikvision Access Control або Honeywell Pro-Watch, які дозволяють обслуговувати тисячі користувачів, підтримують масштабування та централізовану аналітику.

Водночас на ринку існують і відкриті платформи з можливістю розробки власного програмного забезпечення, наприклад OpenAccess чи DoorPi. Вони

використовуються у навчальних і наукових проєктах, де потрібно створити експериментальну модель або налаштувати нестандартну логіку роботи системи.

Загальною тенденцією розвитку сучасних СКУД є перехід до гібридних рішень, які поєднують апаратну автономність із можливістю підключення до хмарних сервісів. Це забезпечує високу надійність системи навіть у разі втрати зв’язку з сервером.

Також активно впроваджуються технології штучного інтелекту — для розпізнавання облич, аналізу поведінкових патернів користувачів та виявлення підозрілих дій. У спортивних комплексах такі системи допомагають забезпечити не лише контроль доступу, а й загальну безпеку на території об’єкта.

Таким чином, сучасні комп’ютерні системи контролю доступу демонструють високий рівень технологічного розвитку, інтеграції та гнучкості. Їхнє впровадження у спортивних комплексах дозволяє автоматизувати управління потоками клієнтів, підвищити безпеку, покращити якість обслуговування та оптимізувати роботу персоналу.

1.3 Класифікація методів ідентифікації користувачів

Ідентифікація користувачів є центральним елементом будь-якої системи контролю доступу. Вона визначає, наскільки точно система може розпізнати особу та надати або обмежити їй доступ до певних зон. У спортивних комплексах, де щодня проходять сотні клієнтів і працівників, важливими є швидкість, надійність і зручність процесу ідентифікації.

Сучасні системи контролю доступу класифікують методи ідентифікації за кількома критеріями: типом інформації, способом розпізнавання, рівнем захищеності та технологічною реалізацією. Традиційно виділяють три базові групи методів — за володінням, за знанням і за біометричними характеристиками. Окрім цього, у практиці сучасних рішень все частіше використовуються методи поведінкової ідентифікації та багатофакторна аутентифікація [6].

Методи ідентифікації за володінням ґрунтуються на наявності у користувача фізичного носія — карти, брелока, браслета або мобільного пристрою. Найпоширенішими є безконтактні картки RFID чи MIFARE, які забезпечують швидке

та зручне проходження через турнікети. Такий підхід зручний і недорогий, однак має недолік — можливість передачі картки іншій особі [6].

Щоб зменшити цей ризик, системи часто поєднують ідентифікацію за володінням з додатковими методами, наприклад введенням PIN-коду. Це дозволяє створити більш безпечну багаторівневу систему контролю.

Ідентифікація за знанням передбачає використання певної секретної інформації, яку знає лише користувач, — пароля, кодового слова або PIN-коду. У спортивних комплексах цей метод застосовується переважно для адміністративного персоналу або при вході в службові приміщення. Його недоліком є необхідність запам’ятовування паролів та ризик їх розголошення [6].

Методи за біометричними характеристиками базуються на унікальних фізіологічних властивостях людини. Серед найпоширеніших — сканування відбитків пальців, розпізнавання обличчя, геометрії руки, райдужної оболонки ока, або навіть голосу. Біометричні технології забезпечують найвищий рівень безпеки, адже «ідентифікатор» не можна втратити чи передати іншій особі [6].

Особливо популярними у спортивних комплексах стали системи розпізнавання обличчя, які працюють швидко та не потребують контакту. Це особливо зручно для проходу великих потоків відвідувачів, а також дозволяє інтегрувати дані з систем відеоспостереження.

Четвертою категорією є поведінкові методи ідентифікації, які базуються на аналізі звичок і характерних дій користувача. Система може розпізнавати людину за динамікою ходи, ритмом натискання клавіш або характером рухів смартфона. Хоча такі рішення поки рідше зустрічаються у спортивних закладах, вони активно розвиваються у сфері цифрової безпеки й можуть стати актуальними у майбутньому [7].

Поведінкова ідентифікація особливо перспективна в контексті мобільних додатків спортивних комплексів, де система може підтвердити особу користувача за його способом використання програми, частотою відвідувань чи характером взаємодії з меню

П’ятим і найбільш сучасним напрямом є багатофакторна аутентифікація (MFA). Вона поєднує два або більше незалежних методи: наприклад, RFID-картку (володіння)

+ PIN-код (знання), або розпізнавання обличчя (біометрія) + мобільний токен (володіння). Такий підхід суттєво підвищує рівень безпеки, оскільки навіть у разі компрометації одного фактора зловмисник не зможе отримати повний доступ [7].

MFA дедалі частіше впроваджується у спортивних комплексах, що мають високі вимоги до контролю персоналу чи фінансових транзакцій — наприклад, при оплаті послуг або доступі до службових приміщень [7].

Крім того, останніми роками набувають популярності мобільні ідентифікаційні технології, що використовують Bluetooth Low Energy (BLE), NFC або QR-коди. Смартфон у цьому випадку виступає у ролі цифрового пропуску. Такі системи зручні, бо не потребують видачі окремих карток, а дані про абонемент і відвідування зберігаються у мобільному додатку.

Сучасні СКУД також активно використовують хмарні сервіси, які дозволяють централізовано зберігати дані користувачів, налаштування доступу та журнали подій. Завдяки цьому адміністратор може віддалено керувати всіма об’єктами спортивної мережі, що підвищує ефективність управління.

Ще однією тенденцією розвитку є поєднання біометричних і поведінкових технологій. Наприклад, система може одночасно розпізнавати обличчя користувача та аналізувати його звички входу — час, місце, тип пристрою. Це дозволяє формувати індивідуальні профілі безпеки. З точки зору надійності, біометричні та багатофакторні системи значно перевищують традиційні методи за володінням. Проте їх вартість і складність реалізації є вищими, тому вибір конкретного рішення залежить від бюджету та вимог спортивного комплексу.

У свою чергу, поведінкові методи можуть слугувати додатковим рівнем перевірки — система порівнює поточні дії користувача з типовими і виявляє аномалії. Це дозволяє своєчасно реагувати на спроби несанкціонованого доступу.

Загалом, ефективна система контролю доступу в спортивному комплексі має поєднувати зручність для користувача з високим рівнем безпеки. Для цього доцільно використовувати комбіновані підходи, що інтегрують декілька методів ідентифікації.

Таким чином, класифікація методів ідентифікації користувачів демонструє широкий спектр технічних можливостей — від класичних RFID-рішень до сучасних

біометричних і поведінкових систем. Вибір конкретного методу визначається призначенням об’єкта, числом користувачів, рівнем конфіденційності інформації та економічними факторами.

Тенденції розвитку у цій галузі свідчать про поступовий перехід до інтелектуальних багатофакторних систем, здатних адаптуватися до поведінки користувача, аналізувати ризики в реальному часі та забезпечувати максимальний рівень безпеки при мінімальному втручанні людини.

1.4. Аналіз вимог до систем контролю доступу у спортивних закладах

У сучасних умовах функціонування спортивних закладів системи контролю доступу (СКУД) є невід’ємним елементом їх інфраструктури. Вони забезпечують безпечне, зручне та ефективне керування потоками відвідувачів, персоналу й обслуговуючих служб. Для створення ефективної системи необхідно чітко визначити вимоги до її функціональності, надійності, масштабованості та взаємодії з іншими підсистемами.

Основною метою впровадження СКУД у спортивних закладах є забезпечення безпеки доступу до приміщень і зон, які мають обмежений вхід. Система повинна гарантувати, що лише авторизовані користувачі можуть потрапити на територію комплексу або в окремі його частини. При цьому важливо забезпечити безперебійне функціонування навіть у години пікового навантаження [8].

До функціональних вимог належить підтримка різних способів ідентифікації — карток, брелоків, біометрії, мобільних пристроїв або комбінації цих методів. Система має бути гнучкою, щоб адміністратор міг самостійно змінювати правила доступу, часові інтервали, рівні прав користувачів та обмеження для певних груп відвідувачів [8].

Важливою характеристикою є інтеграційна здатність системи. У сучасних спортивних комплексах контроль доступу тісно взаємодіє з іншими підсистемами — обліку відвідувань, відеоспостереження, обліку часу роботи персоналу, платіжними та CRM-системами. Така інтеграція дозволяє автоматизувати адміністративні процеси й забезпечити централізоване управління даними [8].

Не менш важливою вимогою є захист персональних даних. Оскільки система працює з особистою інформацією користувачів — ідентифікаторами, фотографіями, біометричними ознаками — вона повинна відповідати вимогам чинного законодавства щодо конфіденційності. Передача даних через мережу має здійснюватися з використанням захищених протоколів шифрування.

З технічного погляду система повинна забезпечувати високу надійність та безперервність роботи. Це передбачає наявність резервного живлення, стійкість до збоїв, можливість відновлення після аварій і дублювання критично важливих компонентів. У спортивних закладах з великою кількістю відвідувачів особливо важливою є пропускна здатність системи. Турнікети, зчитувачі та програмне забезпечення повинні працювати швидко, щоб уникнути черг і затримок під час проходу. Середній час ідентифікації користувача має становити не більше 1–2 секунд. Окремо слід виділити зручність використання для відвідувачів і персоналу. Інтерфейси системи мають бути інтуїтивно зрозумілими, а процес авторизації — максимально простим. Це підвищує лояльність клієнтів і зменшує навантаження на адміністраторів [8].

СКУД повинна також підтримувати гнучке керування правами доступу. Наприклад, клієнт із денним абонементом може відвідувати спортзал лише у певні години, а персонал — у робочий час. Такі налаштування мають задаватися автоматично на основі даних з бази користувачів [9].

Ще одним важливим аспектом є масштабованість системи. Зі зростанням кількості клієнтів чи появою нових залів система повинна дозволяти легко додавати нові точки контролю, турнікети або пристрої без необхідності кардинальної перебудови архітектури.

Серед експлуатаційних вимог варто відзначити простоту обслуговування та оновлення програмного забезпечення. Система має передбачати можливість дистанційного моніторингу стану обладнання, автоматичного сповіщення про несправності та централізованого оновлення компонентів. Особливу увагу слід приділити аналітичним можливостям системи. Сучасні комплекси потребують збору статистики щодо відвідуваності, пікових годин, тривалості перебування клієнтів та

ефективності використання ресурсів. Ці дані можуть бути використані для планування персоналу, маркетингових рішень та оптимізації навантаження [9].

Система має підтримувати автоматичне формування звітів — як для адміністрації, так і для бухгалтерії. Звіти можуть містити дані про кількість відвідувачів, статуси абонементів, час входу й виходу працівників, кількість порушень доступу тощо. З точки зору користувачів, важливою є інтерактивність та цифрова інтеграція. Клієнт повинен мати можливість переглядати свою історію відвідувань, бронювати зали, оплачувати абонементи онлайн. Це створює єдину екосистему обслуговування клієнта. Для адміністрації спортивного закладу СКУД має забезпечувати гнучке управління персоналом. Це включає ведення журналів входів/виходів, облік робочого часу та аналіз дотримання графіка працівниками. Ще однією вимогою є сумісність із мобільними пристроями. Усе більше рішень реалізуються у вигляді мобільних додатків для адміністраторів, що дозволяють дистанційно відстежувати стан системи, блокувати доступ або змінювати налаштування у реальному часі [9].

Також слід враховувати естетичний аспект системи контролю доступу, особливо для сучасних фітнес-клубів і центрів преміум-класу. Обладнання має гармонійно вписуватись у дизайн приміщення та не створювати дискомфорту для відвідувачів.

У деяких випадках системи контролю доступу повинні бути адаптивними до осіб з обмеженими можливостями. Це може включати спеціальні турнікети, знижені зчитувачі або голосові підказки.

Отже, аналіз вимог до систем контролю доступу у спортивних закладах показує, що такі системи мають бути багатофункціональними, інтегрованими, надійними й орієнтованими на комфорт користувача. Їх проектування потребує врахування не лише технічних параметрів, а й організаційних, правових та ергономічних факторів.

2 ОБҐРУНТУВАННЯ ТЕХНОЛОГІЧНОГО СТЕКУ ТА АРХІТЕКТУРИ СИСТЕМИ

2.1 Вибір технологій для розробки мережевих сервісів та веб-інтерфейсу

Розробка комп’ютерної системи контролю доступу до спортивного комплексу вимагає обґрунтованого вибору технологій, які забезпечують ефективну взаємодію між користувачами, апаратними пристроями, базою даних та сервером. У сучасних умовах такі системи мають відповідати ряду ключових вимог: надійність, безпека, масштабованість, інтеграційна сумісність, а також зручність для користувача. Для досягнення цих цілей розробка системи повинна базуватися на сучасному технологічному стеку, який поєднує веб-технології, серверні рішення та засоби обробки запитів у мережевому середовищі.

Основною серверною платформою для розробки системи обрано Node.js. Це середовище виконання JavaScript, побудоване на рушії Google V8, яке забезпечує асинхронну подієву архітектуру. Node.js особливо ефективний у випадках, коли потрібно обробляти численні одночасні підключення користувачів або пристроїв — що саме характерно для систем контролю доступу, де одночасно можуть працювати десятки зчитувачів або турнікетів [10].

Як альтернатива розглядалися платформи Python (Flask, Django) або ASP.NET, однак Node.js забезпечує кращу швидкодію для реальних мережевих сервісів та має зручну інтеграцію з фронтенд-фреймворками.

Для зберігання даних користувачів, подій, журналів доступу та статистики обрано реляційну базу даних MySQL. Вона забезпечує високу надійність, підтримує транзакційність і добре підходить для структурованих даних, таких як записи користувачів, абонементів і логів [11].

Серед переваг MySQL є :

— висока швидкість обробки запитів;

— підтримка складних sql-запитів, зв’язків і тригерів;

— можливість інтеграції з orm-бібліотеками, наприклад sequelize для node.js [11].

У випадках, коли система має потребу в обробці великого обсягу неструктурованих або аналітичних даних (наприклад, логи у форматі JSON або статистика користування), доцільним є використання MongoDB — документоорієнтованої бази даних NoSQL. Вона дозволяє гнучко масштабувати систему, що стане важливою перевагою при розширенні функціональності або інтеграції з іншими сервісами спортивного комплексу.

Комунікація між клієнтською частиною, сервером і апаратними пристроями реалізується через RESTful API, який є стандартом у розробці сучасних вебсистем. REST забезпечує незалежність клієнта та сервера, дозволяючи легко оновлювати систему без зупинки її роботи [12].

API відповідатиме за:

— отримання запитів зчитувачів і турнікетів (наприклад, при проходженні користувача);

— авторизацію та аутентифікацію користувачів;

— обробку статистичних даних;

— надання адміністратору інформаційних звітів через вебінтерфейс [12].

Для швидшої реакції системи у режимі реального часу можна додатково використовувати протокол WebSocket або MQTT, який оптимізовано для IoT-пристроїв. Це дозволяє забезпечити миттєву синхронізацію між контролером і сервером, що особливо важливо при контролі доступу до приміщень.

Оскільки система оперує конфіденційними даними користувачів, надзвичайно важливим є впровадження комплексного захисту. Основні заходи включають використання протоколу HTTPS з SSL/TLS шифруванням для передавання даних, реалізацію JWT (JSON Web Token) для безпечної авторизації користувачів, використання хешування паролів із алгоритмом bcrypt або SHA-256 та контроль доступу на рівні ролей (адміністратор, інструктор, відвідувач).

Такі заходи дозволять мінімізувати ризики несанкціонованого доступу, витоку даних або атак типу «людина посередині» [12].

Розглянемо технології для веб-інтерфейсу. Інтерфейс системи має бути зрозумілим, швидким та адаптивним. Для цього обрано React.js — сучасну бібліотеку для створення односторінкових вебдодатків (SPA — Single Page Application). React дозволяє створювати динамічні інтерфейси, що швидко оновлюються без перезавантаження сторінки. Це забезпечує комфортну роботу адміністратора, який може в реальному часі переглядати стан турнікетів, логіни користувачів, кількість присутніх у залі та статистику відвідувань [12].

Додатково слід використати технології:

— Bootstrap 5 або Tailwind CSS — для створення сучасного адаптивного дизайну;

— Axios — для взаємодії з REST API;

— Chart.js або Recharts — для побудови графіків і статистики;

— Redux або Zustand — для централізованого керування станом застосунку.

Для розгортання системи використовується Docker, який дозволяє ізолювати кожен компонент (сервер, база даних, вебінтерфейс) у власному контейнері. Це гарантує стабільну роботу незалежно від середовища, спрощує оновлення та масштабування.

У якості системи контролю версій застосовується Git з хостингом репозиторію на GitHub або GitLab, що дає змогу ефективно координувати командну розробку та впроваджувати оновлення.

Для підвищення ефективності роботи комп’ютерної системи контролю доступу до спортивного комплексу доцільним є використання хмарних технологій. Вони забезпечують стабільне зберігання даних, гнучке масштабування, централізоване адміністрування та можливість швидкої інтеграції з іншими онлайн-сервісами.

Розглянемо таблицю 2.1, яка містить порівняльний аналіз трьох основних рішень — Firebase, AWS (Amazon Web Services), Microsoft Azure, а також додаткових сервісів інтеграції — Telegram API і Email SMTP.

Таблиця 2.1 — Порівняльний аналіз хмарних технологій

Сервіс	Основне призначення	Переваги	Недоліки	Оптимальне застосування
Firebase	Зберігання журналів, аутентифікація, аналітика	Простота, безкоштовність, інтеграція з веб-додатками	Обмежене масштабування	Невеликі системи або пілотні проекти
AWS	Хостинг, обробка, аналітика, IoT	Масштабованість, безпека, універсальність	Висока вартість	Великі системи з високим навантаженням
Azure	Хостинг, аналітика, інтеграція з Microsoft	Безпека, корпоративна інтеграція	Складна конфігурація	Спортивні мережі або корпоративні комплекси
Telegram API	Сповіщення в реальному часі	Швидкість, безкоштовність, інтерактивність	Залежність від месенджера	Оперативні повідомлення персоналу
Email SMTP	Надсилання звітів, статистики	Універсальність, надійність	Повільна доставка	Формальна звітність, щоденні підсумки

Провівши порівняльний аналіз основних хмарних платформ і сервісів, можна зробити висновок, що найбільш доцільним вибором для реалізації комп’ютерної системи контролю доступу до спортивного комплексу є використання зв’язки Firebase + Telegram API.

Firebase забезпечує зручний механізм резервного зберігання даних, автентифікації користувачів, реального часу оновлення подій і хмарних функцій без необхідності адміністрування серверів. Це дозволяє значно скоротити час розробки та спростити підтримку системи, що є важливим у межах невеликих або середніх спортивних закладів [13].

Використання Telegram API як засобу сповіщення адміністратора про підозрілі дії чи спроби несанкціонованого доступу є ефективним рішенням через його високу

швидкість, надійність і зручність у користуванні. Крім того, можливість створення ботів дозволяє реалізувати інтерактивну взаємодію між системою та персоналом.

У випадку, якщо передбачається масштабування проєкту для мережі спортивних комплексів або інтеграція з іншими корпоративними сервісами, доцільним буде розглянути перехід на AWS або Azure, які мають розширені можливості керування навантаженням, безпекою та аналітикою.

Отже, оптимальною конфігурацією для даної системи є:

— Firebase, як основна хмарна платформа для зберігання, синхронізації та автентифікації;

— Telegram API, як засіб оперативного інформування адміністратора про події безпеки [13].

Такий вибір забезпечує баланс між функціональністю, вартістю та простотою впровадження, роблячи систему ефективною, гнучкою та готовою до подальшого розвитку.

Таким чином, на основі аналізу технічних вимог та можливостей сучасних технологій, оптимальний стек для розробки комп'ютерної системи контролю доступу до спортивного комплексу включає:

— Node.js — серверна платформа;

— React.js — клієнтський інтерфейс;

— MySQL або MongoDB — база даних;

— REST API / MQTT — мережевий протокол взаємодії;

— SSL/TLS, JWT — засоби безпеки;

— Git — інструменти розгортання та підтримки.

Поєднання цих технологій дозволить створити сучасну, безпечну та ефективну систему, яка відповідатиме потребам спортивних закладів і забезпечуватиме стабільну роботу в умовах великого навантаження.

2.2 Архітектура та структура комп'ютерної системи

Архітектура системи контролю доступу до спортивного комплексу є ключовим елементом, що визначає її ефективність, масштабованість та безпеку. Вона має забезпечувати надійну взаємодію між апаратною частиною (зчитувачами, контролерами, турнікетами) та програмним забезпеченням, яке здійснює ідентифікацію користувачів, управління правами доступу та обробку подій. Система побудована на принципах багаторівневої клієнт-серверної архітектури, що дозволяє гнучко розподіляти навантаження між компонентами, забезпечувати централізований контроль і можливість розширення.

2.2.1 Багаторівнева архітектура системи контролю доступу

Архітектура системи контролю доступу до спортивного комплексу базується на принципах модульності та багаторівневості, що дозволяє забезпечити надійну роботу, легке масштабування та централізоване управління. Система поєднує апаратні пристрої, програмні модулі та мережеві сервіси, які взаємодіють між собою через стандартизовані інтерфейси [14].

Для ефективного функціонування система поділяється на три рівні:

— фізичний рівень (апаратна частина), що включає пристрої для ідентифікації користувача, збору та виконання команд;

— серверний рівень (логічна обробка даних) для управління, обліку, аналітики, зберігання інформації;

— клієнтський рівень (веб-інтерфейс), як взаємодія з адміністратором і користувачем.

Розглянемо детальніше апаратну частину. На цьому рівні розміщені пристрої, що здійснюють ідентифікацію, контроль і фізичне обмеження доступу до приміщень спортивного комплексу.

Контролер доступу — ZKTeco InBio160 (рисунок 2.1). Це центральний елемент фізичного рівня, який координує роботу всіх периферійних пристроїв. Контролер підключається до серверної частини через Ethernet (TCP/IP) або RS-485, що забезпечує

гнучкість розміщення обладнання. Підтримує до 2 точок доступу, може керувати двома дверима одночасно. Має вбудовану пам’ять на 30 000 користувачів і понад 100 000 записів подій. Працює як у онлайн, так і в офлайн режимі — у разі втрати з’єднання із сервером рішення про доступ приймається локально на основі збережених даних. Має інтегровану підтримку зчитувачів RFID, MIFARE та Wiegand, що дає змогу легко розширювати систему. Використовує програмне забезпечення ZKBioSecurity або сумісні системи для централізованого управління користувачами, розкладом доступу, журналами подій [15].

Рисунок 2.1 — Контролер доступу ZKTeco InBio160

Обрання моделі ZKTeco InBio160 зумовлене її високою надійністю, підтримкою мережевої інтеграції, розширюваністю та наявністю офіційної технічної підтримки. Крім того, пристрій має доступну ціну та легко інтегрується із веб-інтерфейсом через REST API або SDK [15].

Зчитувачі карток RFID (модель ZKTeco KR500E) зображений на рисунку 2.2. Зчитувачі призначені для ідентифікації користувачів за допомогою безконтактних RFID-карт або брелоків. Технологія 125 кГц EM-Marine забезпечує швидке зчитування на відстані до 10 см [16].

Основні характеристики:

— сумісність із більшістю контролерів стандарту Wiegand 26/34;

— високий рівень захисту корпусу (IP65), що дозволяє встановлення як у приміщенні, так і на вулиці;

— індикація у вигляді світлодіодів та звукового сигналу для підтвердження ідентифікації;

— можливість підключення кількох зчитувачів до одного контролера для забезпечення контролю входу та виходу [16].

Рисунок 2.2 — Зчитувач безконтактних карт ZKTeco KR503e

Такий вибір обґрунтований простотою монтажу, сумісністю з контролером InBio160 та високою швидкістю ідентифікації, що особливо важливо у місцях з великим потоком відвідувачів — наприклад, у фітнес-залі або басейні [16].

Електромагнітні замки (модель YLI YM-280N) – замки, що встановлюються безпосередньо на дверях і отримують сигнали від контролера. Сила утримання — 280 кг, що забезпечує надійне блокування. Працюють від 12 В постійного струму, споживання — близько 500 мА. Оснащені індикатором стану (відкрито/закрито). У разі

зникнення живлення спрацьовує режим «Fail Safe» — двері автоматично розблоковуються, що відповідає вимогам пожежної безпеки [17].

Застосування саме цієї моделі виправдане поєднанням надійності, низького рівня шуму та довговічності (понад 500 000 циклів роботи).

Кнопка виходу (Exit Button) та датчики стану дверей

Для зручності користувачів у систему інтегровано кнопку виходу з підсвіткою, що подає сигнал на контролер про відкриття дверей.

Датчик стану дверей контролює, чи двері залишились відчиненими, передаючи цю інформацію на сервер. Це дозволяє системі автоматично формувати сповіщення у випадку несанкціонованого доступу або залишення дверей відкритими понад встановлений час.

Джерело безперебійного живлення (UPS, APC Back-UPS 650VA) для забезпечення безперервної роботи системи контролю доступу використовується UPS, який підтримує живлення контролера, замків і мережевого обладнання у разі зникнення електроенергії. Час автономної роботи — до 30 хвилин, що достатньо для збереження усіх даних і безпечного завершення процесів. Має захист від перенапруги та короткого замикання, що підвищує загальну надійність системи [18].

Мережеве обладнання (комутатор та маршрутизатор)

Для об’єднання всіх пристроїв у єдину мережу використовується комутатор TP-Link TL-SG108 та маршрутизатор TP-Link Archer AX23, які забезпечують стабільне з’єднання з сервером і підтримку протоколів безпеки WPA3.

Рисунок 2.3 — Комутатор TP-Link TL-SG108 та маршрутизатор TP-Link Archer AX23

Коли користувач підносить RFID-картку до зчитувача, відбувається процес радіочастотної ідентифікації. У пам’яті кожної картки зашифрований унікальний код (ID), який є аналогом «ключа» користувача. Зчитувач, використовуючи електромагнітне поле на частоті 125 кГц, активує мікрочіп картки та приймає від неї сигнал. Отриманий ідентифікатор передається через інтерфейс Wiegand або RS-485 до контролера InBio160, який виступає центральним елементом локального вузла доступу.

Контролер отримує цей код і порівнює його з інформацією, що зберігається у внутрішній пам’яті пристрою або в централізованій базі даних сервера. Якщо контролер працює в офлайн-режимі, перевірка здійснюється локально — це дозволяє системі продовжувати функціонування навіть при тимчасовій відсутності зв’язку із сервером. У режимі онлайн контролер надсилає запит через локальну мережу на серверну частину, де програмне забезпечення аналізує, чи має користувач право доступу до певної зони, у визначений час і день.

Після позитивної перевірки система видає команду на вихідний реле контролера, яке подає напругу на електромагнітний замок. Замок моментально розблоковується, дозволяючи користувачу відчинити двері. Якщо ж ідентифікація неуспішна (наприклад, користувач не має відповідних прав або картка не зареєстрована в системі), контролер не подає сигнал на замок, і доступ залишається заблокованим. У цей момент може активуватися звукова або світлова сигналізація, що інформує про невдалу спробу доступу.

Після завершення кожної події — незалежно від її результату — контролер автоматично формує журнал подій, у якому фіксуються: унікальний ідентифікатор користувача, час і дата проходу, результат (доступ дозволено або відмовлено), місце (номер турнікета, дверей або зони), технічні повідомлення (помилка зчитування, втрата зв’язку тощо).

Ці дані надсилаються на серверну базу даних, де зберігаються в хронологічному порядку. Серверна частина системи, використовуючи програмне забезпечення, виконує аналітику журналів — може генерувати звіти, виявляти підозрілі дії (наприклад, багаторазові спроби входу з однією карткою) та надсилати повідомлення адміністратору через Telegram API або Email SMTP.

Адміністратор, авторизувавшись у веб-інтерфейсі системи, має змогу в режимі реального часу бачити всі події, отримувати інформацію про поточний стан дверей (відкрито/закрито), активність користувачів та з’єднання контролерів. Усі дані в інтерфейсі візуалізуються у вигляді таблиць, графіків і статистичних панелей, що полегшує моніторинг і керування комплексом.

Важливо, що архітектура передбачає двосторонній обмін інформацією — адміністратор може не лише переглядати дані, а й віддалено блокувати користувача, змінювати його права доступу або відкривати двері вручну у разі надзвичайних ситуацій (наприклад, пожежі чи евакуації).

Таким чином, описаний алгоритм демонструє, що система функціонує як замкнений цикл зворотного зв’язку: від моменту ідентифікації користувача до запису події в базу даних і відображення інформації у веб-інтерфейсі. Це забезпечує повну прозорість і контроль над усіма процесами доступу в спортивному комплексі.

Завдяки такій побудові, фізичний рівень системи є надійною базою для всієї багаторівневої архітектури, адже саме він забезпечує реальну взаємодію користувача з технічними засобами. У додатку В зображено схему багаторівневої архітектури управління доступом та даними в хмарному середовищі, з використанням механізмів шифрування та політик доступу.

Використання сучасних RFID-зчитувачів, високоякісних контролерів та серверного ПЗ дає змогу досягти високого рівня безпеки, стабільності й масштабованості.

Обрані прилади — InBio160, KR500E, YLI YM-280N та UPS APC 650VA — утворюють оптимальний комплекс за співвідношенням функціональності, вартості, енергоефективності та сумісності, що робить їх доцільним вибором для впровадження у сучасному спортивному закладі будь-якого рівня — від приватного фітнес-центру до великого муніципального спортивного комплексу [18].

2.2.2 Схема взаємодії між апаратним забезпеченням та мережевими сервісами

У сучасних системах контролю доступу, зокрема у спортивних комплексах, надзвичайно важливим є чітка взаємодія між апаратними компонентами та мережевими сервісами. Саме така взаємодія забезпечує не лише фізичний контроль

над входом і виходом користувачів, а й централізований збір, обробку та аналіз даних про всі події, що відбуваються в системі. Архітектура проєктованої системи передбачає послідовну роботу кількох взаємопов’язаних рівнів: апаратного, мережевого, серверного, хмарного та прикладного. Кожен із цих рівнів виконує свою функцію, але лише у поєднанні вони формують єдину, надійну інформаційну екосистему.

На базовому рівні функціонує апаратна частина, яка відповідає за безпосередній контакт із користувачем. До неї належать RFID-зчитувачі, контролери, електромагнітні замки, турнікети, датчики відкриття дверей і сигналізації. Коли користувач підносить картку до зчитувача, відбувається передача унікального коду через стандартний інтерфейс (наприклад, Wiegand або RS-485) до контролера. Саме контролер — у нашому випадку модель InBio160 — виступає ключовою ланкою, яка приймає рішення про дозвіл чи заборону доступу. Він аналізує отриманий ідентифікатор, звертається до бази даних користувачів, виконує перевірку правил і приймає рішення про відкриття замка [19]. Блок-схема апаратної частини системи контролю доступу на базі RFID зображена у додатку Г.

На наступному рівні працює локальна мережа, через яку контролери обмінюються інформацією з центральним сервером. Комунікація здійснюється за протоколом TCP/IP, що дозволяє інтегрувати систему у стандартну мережеву інфраструктуру спортивного комплексу. Для забезпечення стабільності передача даних може бути захищена за допомогою SSL/TLS-шифрування, яке унеможливлює втручання третіх осіб у процес автентифікації або зміну даних під час передачі. За потреби система може підтримувати резервні канали зв’язку, наприклад, через Wi-Fi або GSM-модем, що гарантує безперервність роботи навіть при аварійних відключеннях [19].

Центральним елементом системи є сервер контролю доступу, який виконує обробку запитів, управління правами користувачів і ведення журналів подій. У ньому зберігається база даних, де міститься повна інформація про всіх користувачів, зони доступу, розклади, ролі та привілеї. Сервер приймає запити від контролерів, здійснює перевірку ідентифікатора та повертає відповідь: "доступ дозволено" або "відмовлено".

Крім того, сервер формує статистичні звіти, веде журнал усіх спроб входу, ідентифікує підозрілі події, такі як багаторазові спроби несанкціонованого входу.

Для підвищення стійкості система використовує хмарні сервіси, які виконують роль резервного сховища та платформи для масштабування. Наприклад, сервіс Firebase може зберігати копії журналів подій у реальному часі, що забезпечує збереження інформації навіть у разі збоїв основного сервера. Платформи AWS або Microsoft Azure дозволяють розмістити серверну частину у хмарі, забезпечуючи відмовостійкість, швидке масштабування та централізований моніторинг. Таке рішення робить систему більш гнучкою — доступ до неї може здійснюватися з будь-якої точки світу, а адміністратор має змогу контролювати всі об’єкти спортивного комплексу дистанційно.

Важливою складовою є також механізми сповіщення. За допомогою Telegram API або Email SMTP система може автоматично надсилати повідомлення адміністраторам чи охороні про критичні події — наприклад, спроби входу заблокованого користувача, порушення графіка роботи або відключення контролера від мережі. Це забезпечує оперативне реагування та підвищує рівень безпеки.

На прикладному рівні працює веб-інтерфейс адміністратора — інтерактивна панель, створена з використанням сучасних технологій (HTML5, CSS3, JavaScript, React). Через цей інтерфейс адміністратор має змогу бачити всі події у режимі реального часу, додавати нових користувачів, змінювати їхні права доступу, відкривати двері дистанційно або блокувати підозрілі облікові записи [19].

Взаємодія між усіма компонентами відбувається за принципом "клієнт–сервер–хмара". Кожен запит на вхід ініціює короткий, але насичений обмін даними: зчитувач передає ідентифікатор → контролер обробляє або надсилає його на сервер → сервер повертає результат → контролер приймає рішення → подія фіксується у базі даних та дублюється в хмарі. Водночас адміністратор отримує інформацію у своєму веб-інтерфейсі практично миттєво (рисунок 2.4).

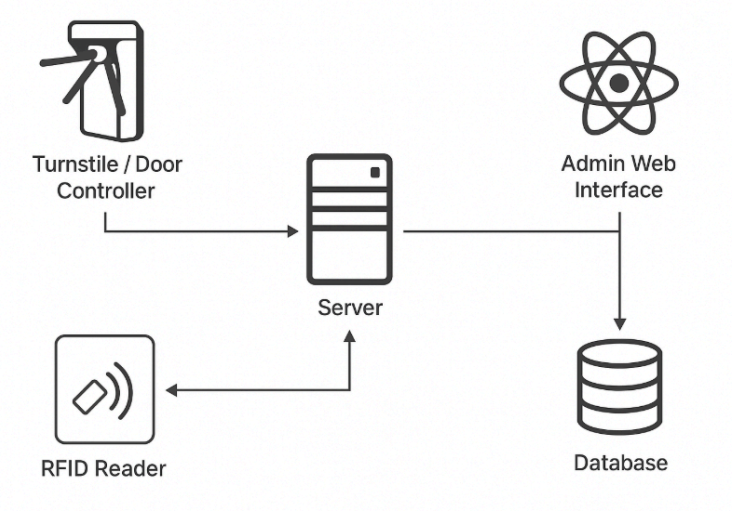

Рисунок 2.4 — Схема взаємодії між апаратним забезпеченням та мережевими сервісами

Така структура дозволяє досягти високого рівня узгодженості та безпеки. Якщо з якоїсь причини втрачається з’єднання з сервером, контролери продовжують працювати автономно, використовуючи локальні копії бази даних. Після відновлення зв’язку всі події автоматично синхронізуються з центральним сховищем. Таким чином, система є не лише функціональною, а й надійною у технічному плані [20].

У результаті формується інтелектуальна екосистема, де апаратне обладнання, програмне забезпечення та хмарні технології діють як єдиний організм. Такий підхід забезпечує не лише фізичну безпеку спортивного комплексу, а й ефективне управління його інфраструктурою, спрощуючи контроль, облік і моніторинг. Це робить систему сучасною, масштабованою та придатною до впровадження у будь-яких умовах, від невеликих фітнес-центрів до багатоповерхових спортивних арен [20].

2.2.3 Модель локалізації в бездротових мережах на базі RSSI-показників

У межах системи контролю доступу важливою задачею є визначення просторового положення користувача всередині спортивного комплексу. Це дозволяє здійснювати моніторинг переміщень, підвищувати безпеку, оптимізувати навантаження на тренувальні зони та забезпечувати додаткову аналітику. Одним із найбільш доступних та універсальних підходів є використання індикатора потужності

прийнятого сигналу – RSSI (Received Signal Strength Indicator), що передається від персонального ідентифікатора (BLE-мітка, смартфон) до локальних приймачів.

RSSI-сигнали характеризуються суттєвими флуктуаціями, зумовленими особливостями поширення радіохвиль у приміщенні: багатопроменевістю, поглинанням матеріалами, екрануванням тілами відвідувачів тощо. Тому для отримання стабільних параметрів положення необхідно застосовувати математичні моделі, які компенсують ці спотворення.

Базовий зв'язок між RSSI та відстанню до приймача описується моделлю логарифмічного загасання:

$$d = d0 * 10^{((RSSI0 - RSSI) / (10 * n))},$$

де d — оцінена відстань до передавача;

d0 — базова відстань калібрування (1 м);

RSSI0 — середня потужність сигналу на відстані d0;

RSSI — поточне значення сигналу;

n — коефіцієнт загасання (2–4 у приміщенні).

Оскільки RSSI містить шум, застосовується фільтрація. Часто використовують експоненційне згладжування:

$$RSSI\_filtered(t) = \alpha * RSSI(t) + (1 - \alpha) * RSSI\_filtered(t - 1),$$

де α — коефіцієнт згладжування (0.1–0.3).

Після оцінки відстаней до кількох приймачів можна перейти до трилатерації. Нехай існує n приймачів із координатами (xi, yi). Тоді для кожної точки виконується:

$$(x - xi)^2 + (y - yi)^2 = di^2$$

Оскільки вимірювання мають шум, система є надлишковою і розв’язується методом найменших квадратів:

$$\min \Sigma ((x - xi)^2 + (y - yi)^2 - di^2)^2.$$

Другий підхід — fingerprinting (радіокарта). У цьому методі приміщення попередньо поділяють на контрольні точки. У кожній точці знімають середній RSSI від n приймачів, утворюючи вектор:

$$r_x = (r1, r2, \ldots, rn)$$

Для нового вимірювання r система порівнює його з усіма еталонними точками і знаходить найближчу за метрикою, наприклад, Евкліда:

$$d(r, r_x) = ||r - r_x||_2$$

Додатково можна працювати з небалансованими або нестабільними сигналами, застосовуючи:

— нормалізацію RSSI;

— обчислення вагів для кожного приймача;

— детектор викидів значень;

— оцінку на основі стохастичної моделі.

Комбінована модель (трилатерація + fingerprinting) поєднує переваги обох підходів: трилатерація дає грубу оцінку, а fingerprinting виконує уточнення.

Локалізаційний модуль інтегрується в архітектуру системи як окремий сервіс, що отримує потоки RSSI від контролерів, обробляє їх у режимі реального часу та передає координати до модуля моніторингу та аналітики. Таким чином забезпечується можливість побудови теплових карт завантаженості, відстеження переміщень та

оперативного реагування на нестандартні ситуації. У додатку Д зображено UML-модель процесу позиціонування.

2.3 Проектування структури бази даних

Проєктування бази даних є одним із ключових етапів розроблення комп’ютерної системи контролю доступу, оскільки саме вона забезпечує збереження, цілісність і швидкий доступ до інформації про користувачів, події, рівні доступу та налаштування системи. Грамотно побудована структура бази даних дозволяє оптимізувати роботу системи, підвищити її надійність, а також забезпечити можливість масштабування у майбутньому.

Під час проєктування бази даних системи опрацювання сигналів електричних перетворювачів руху було враховано низку ключових принципів, що забезпечують ефективність, надійність і гнучкість роботи всієї системи. Одним із головних принципів стала нормалізація даних, яка передбачала усунення дублювання інформації шляхом поділу її на взаємопов’язані таблиці. Такий підхід дозволив мінімізувати надлишковість даних, підвищити узгодженість записів і спростити подальше оновлення або розширення структури бази [21].

Не менш важливою складовою є забезпечення цілісності даних. Для цього в структурі бази були реалізовані первинні та зовнішні ключі, що гарантують коректність зв’язків між таблицями та запобігають появі неконсистентних записів. Це дало змогу зберегти логічну узгодженість інформації навіть при одночасному доступі кількох користувачів або під час виконання складних транзакцій [21].

Особливу увагу приділено питанням безпеки, адже система може містити технічні та службові дані, що потребують обмеженого доступу. Було впроваджено контроль прав користувачів, який визначає рівень доступу до таблиць і операцій із даними. Це забезпечує захист від несанкціонованих змін або втрати важливої інформації [21]. Також під час розроблення враховано принцип масштабованості, що передбачає можливість розширення бази даних у майбутньому. Архітектура системи дозволяє без складних змін додавати нові функціональні модулі — наприклад, підсистему аналітики, статистики чи біометричної ідентифікації.

Для зберігання даних використано реляційну базу даних, MySQL, так як вона підходить для роботи з вебсервером (Node.js, Django, Flask тощо) і підтримує високий рівень безпеки та гнучкість у запитах.

Логічна структура бази даних системи контролю доступу містить кілька основних сутностей, які відображають реальні об’єкти системи: користувачів, пристрої, зони доступу, журнали подій тощо. Нижче наведено основні таблиці та їх призначення.

Таблиця Users — містить інформацію про всіх зареєстрованих осіб (відвідувачів, тренерів, персонал). Основні її поля:

— user_id — унікальний ідентифікатор користувача (PRIMARY KEY);

— full_name — ПІБ користувача;

— rfid_code — унікальний код картки або брелка;

— biometric_id — хеш або ідентифікатор біометричних даних (якщо використовується);

— access_level_id — зовнішній ключ до таблиці рівнів доступу;

— registration_date — дата реєстрації;

— status — активний/заблокований.

Таблиця AccessLevels — Мвизначає права відвідування певних зон комплексу. Містить поля:

— access_level_id — ідентифікатор рівня (PRIMARY KEY);

— level_name — назва (наприклад: “Абонемент стандарт”, “Тренер”, “Адміністратор”);

— description — текстовий опис доступу;

— time_limits — часові обмеження (наприклад, дозволений доступ з 8:00 до 22:00).

Таблиця Zones — описує приміщення або частини спортивного комплексу. Поля:

- zone_id — унікальний ідентифікатор;
- zone_name — назва зони (“Тренажерний зал”, “Басейн”, “Сауна”);
- controller_id — контролер, який обслуговує цю зону;
- description — додаткова інформація.

Таблиця Controllers — визначає апаратні пристрої системи (InBio160 тощо). Поля:

- controller_id — ідентифікатор пристрою (PRIMARY KEY);
- ip_address — IP-адреса контролера;
- location — фізичне розташування;
- status — активний/відключений;
- last_connection — дата та час останнього зв’язку.

Таблиця Logs — містить записи про всі спроби доступу. Включає в себе поля:

- log_id — унікальний ідентифікатор події;
- user_id — користувач, який намагався отримати доступ;
- zone_id — зона або двері, до яких здійснювався доступ;
- event_type — тип події (“доступ дозволено”, “відмова”, “помилка”);
- timestamp — дата й час події;
- device_source — зчитувач або контролер, який зафіксував подію.

Зв’язки між таблицями в базі даних системи контролю доступу побудовані таким чином, щоб забезпечити цілісність, логічну узгодженість та зручність подальшої аналітики даних. База даних розроблена за реляційним принципом, де кожна таблиця має чітке функціональне призначення, а зв’язки між ними реалізуються через зовнішні ключі (foreign keys). Це дозволяє уникнути дублювання даних і гарантує, що будь-яка

дія користувача, зафіксована у системі, може бути простежена до конкретного об’єкта, контролера та зони.

Першим і основним зв’язком **є** відношення між таблицями Users та AccessLevels**.** Кожен користувач системи має певний рівень доступу, який визначає, у які зони він може потрапити, у який час та з якими правами. Зв’язок між цими таблицями має тип «один до багатьох», тобто один запис у таблиці AccessLevels може відповідати багатьом користувачам, але кожен користувач має лише один рівень доступу. Такий підхід забезпечує централізоване керування політиками безпеки: змінивши параметри рівня доступу, адміністратор автоматично оновлює права для всіх користувачів цього рівня.

Другий ключовий зв’язок — між таблицями Controllers та Zones.
Контролер відповідає за фізичне керування дверима або турнікетами у певних зонах. Один контролер може обслуговувати декілька зон, наприклад, різні входи або приміщення спортивного комплексу. Цей зв’язок також має тип «один до багатьох»: один контролер — багато зон. Така структура забезпечує масштабованість системи: при розширенні комплексу можна додавати нові зони, не змінюючи загальну логіку роботи контролера.

Наступний важливий зв’язок — між таблицями Users та Logs.
Таблиця Logs містить усі записи про події — зчитування карток, дозволені чи заборонені спроби доступу, час входу і виходу користувача. Один користувач може створювати необмежену кількість записів у журналі, тому цей зв’язок також є «один до багатьох». Така структура дає змогу вести історію дій кожного користувача, аналізувати активність і виявляти потенційні порушення безпеки.
Додатково, таблиця Logs пов’язана з таблицями Zones та Controllers.
Кожен запис у журналі подій містить ідентифікатор зони, у якій відбулася дія, та ідентифікатор контролера, що її обслуговує. Це забезпечує повну трасування подій: можна відстежити, який користувач, коли і через який контролер отримав (або не отримав) доступ до конкретної зони.

Таким чином, усі зв’язки між таблицями формують єдину логічну модель системи, де Users**,** AccessLevels**,** Controllers**,** Zones **і** Logs взаємодіють між собою через

зовнішні ключі. Це дозволяє зберігати дані у нормалізованому вигляді, зменшити ризик помилок при оновленні інформації та спростити аналітичну обробку.

Для підвищення продуктивності передбачено:

— створення індексів для полів user_id, rfid_code, timestamp;

— використання триггерів для автоматичного створення записів у Logs;

— регулярне резервне копіювання через хмарні сервіси (Firebase, AWS S3);

— обмеження доступу до бази через ролі користувачів (адміністратор, оператор, гість).

Для безпеки даних усі паролі та біометричні ідентифікатори зберігаються у зашифрованому вигляді (алгоритми SHA-256, bcrypt тощо). Передбачено використання SSL-з'єднання для передачі даних між сервером і клієнтом.

Таким чином, розроблена структура бази даних забезпечує логічну цілісність, гнучкість та високу швидкість обробки інформації. Вона дозволяє ефективно зберігати дані про користувачів, контролювати доступ до різних зон спортивного комплексу, вести історію подій і здійснювати моніторинг у реальному часі.

Архітектура бази є модульною, тому її можна легко розширити, додавши, наприклад, модуль аналітики відвідуваності чи інтеграцію з системою оплати абонементів. ER-діаграма структури бази даних представлена у додатку Е.

2.4 Проєктування REST API для обміну даними

У процесі створення комп'ютерної системи контролю доступу важливим етапом є проєктування інтерфейсу прикладного програмування — REST API (Representational State Transfer Application Programming Interface). Саме цей компонент забезпечує інформаційну взаємодію між апаратними пристроями, серверною частиною системи, базою даних і клієнтськими застосунками. REST API виступає центральною ланкою, через яку здійснюється обмін даними між контролерами, зчитувачами, адміністративною панеллю та аналітичними сервісами, що забезпечує цілісність, узгодженість і актуальність усіх процесів у системі [22].

REST API ґрунтується на принципі репрезентації ресурсів, де кожен сутнісний об’єкт системи (користувач, зона, контролер, журнал подій) подається як окремий ресурс із власним унікальним ідентифікатором. Комунікація між клієнтом і сервером відбувається через стандартні HTTP-запити з використанням методів GET, POST, PUT, PATCH та DELETE. Такий підхід дозволяє чітко структурувати логіку запитів, зробити систему розширюваною та зручною для інтеграції з іншими компонентами, зокрема мобільними застосунками або зовнішніми аналітичними платформами [22].

У проєктованій системі REST API виконує декілька критично важливих функцій. По-перше, він забезпечує передачу даних від апаратного рівня (контролери доступу, зчитувачі, замки) до серверної частини, де ці дані зберігаються, аналізуються та відображаються в інтерфейсі адміністратора. По-друге, API дозволяє виконувати зворотну комунікацію — тобто відправляти з сервера до пристроїв команди оновлення прав доступу, синхронізації даних або блокування користувача. По-третє, API реалізує механізми аналітики, формування звітів і сповіщень у реальному часі, що робить систему не лише інструментом контролю, а й інтелектуальною платформою управління доступом [22].

Одним із ключових аспектів проєктування API є забезпечення безпеки. У запропонованій архітектурі всі запити відбуваються виключно через захищений протокол HTTPS, що запобігає перехопленню або підробці даних під час передачі. Для автентифікації користувачів використовується механізм токенів (JWT — JSON Web Token), який гарантує, що доступ до ресурсів мають лише авторизовані суб’єкти. Адміністративні користувачі отримують короткострокові токени для роботи у веб-інтерфейсі, а контролери і зчитувачі — довготривалі ключі доступу або сертифікати, які зберігаються в їхній пам’яті. Це дозволяє будувати систему безпеки на рівні не лише програмного, а й апаратного контролю, запобігаючи спробам несанкціонованого доступу або фальсифікації подій [23].

З архітектурного погляду REST API є stateless-сервісом, тобто він не зберігає даних про попередні запити користувача. Кожен запит розглядається як незалежний, що дозволяє масштабувати сервер горизонтально та забезпечує стабільну роботу при великій кількості одночасних підключень. Такий підхід особливо важливий у

спортивних комплексах, де в години пікового навантаження (наприклад, перед початком тренувань або матчів) система може обробляти десятки тисяч звернень від контролерів та користувачів [23].

Проєктування REST API передбачає чітке визначення основних ресурсів, які відповідають сутностям бази даних. До них належать користувачі, рівні доступу, контролери, зони та журнали подій. Для кожного ресурсу визначаються стандартні операції — отримання списку, перегляд конкретного елемента, створення, оновлення та видалення. Наприклад, запит GET /api/v1/users повертає список усіх зареєстрованих користувачів, тоді як POST /api/v1/users створює нового користувача в системі з визначеним рівнем доступу. Така структурованість дозволяє забезпечити однакову логіку обробки запитів, спрощує супровід коду та підвищує узгодженість між різними модулями системи [23].

Особливу увагу під час проєктування приділяють формату передавання даних. У системі використовується формат JSON, який є легким для обробки як на стороні сервера, так і на клієнтських пристроях. Наприклад, під час події зчитування RFID-картки контролер формує повідомлення, яке містить ідентифікатор пристрою, код картки, час події та статус доступу, після чого надсилає його до сервера. Сервер, отримавши запит, перевіряє відповідність даних у базі та приймає рішення про надання або відмову у доступі. Усі події фіксуються в таблиці журналу, що забезпечує можливість подальшого аудиту.

Важливим елементом REST API є механізм валідації даних. Перед збереженням будь-якої інформації система перевіряє правильність структури запиту, відповідність обов’язкових полів, формат часових позначок та коректність ідентифікаторів. Це дозволяє уникнути помилок при обміні даними між різними компонентами та гарантує стабільність системи навіть за великої кількості одночасних транзакцій.

Оскільки система контролю доступу є критично важливою з точки зору безпеки, REST API має також реалізовувати механізми моніторингу та логування. Кожен запит з боку контролера або адміністратора реєструється в окремому журналі API-звернень із зазначенням часу, типу операції, IP-адреси клієнта та результату виконання. У разі виникнення помилок система автоматично надсилає сповіщення адміністратору, що

дозволяє оперативно реагувати на збої або підозрілі активності. Додатково передбачено функцію обмеження частоти запитів (rate limiting), яка захищає сервер від перевантаження або атак типу “відмова в обслуговуванні” [24].

У проєктованій системі REST API виступає не лише технічним інструментом зв’язку, а й концептуальною основою для побудови всієї архітектури. Його універсальність, гнучкість і безпечність роблять можливим поєднання різних компонентів — від фізичних пристроїв до аналітичних панелей у веб-інтерфейсі. Впровадження такого API забезпечує централізоване управління доступом, швидку обробку запитів, зручне адміністрування та високий рівень надійності системи в цілому [24].

Перейдемо до реалізації REST API, починаємо з формалізації основних сценаріїв використання системи та переліку ресурсів, які ці сценарії оперують. Для системи контролю доступу спортивного комплексу практично визначальними є такі сценарії: реєстрація та оновлення даних користувача і абонемента; ідентифікація при проході через зчитувач; синхронізація локальної бази контролера із сервером; централізований збір журналів подій; вручне або автоматичне блокування доступу у разі прострочення абонемента; генерація аналітичних звітів та сповіщення адміністратора про інциденти. Кожний із цих сценаріїв лягає в основу конкретних ендпоінтів API і визначає формат обміну повідомленнями.

Архітектурно API має бути розбитим на версії (наприклад, /api/v1/...), що дозволить у майбутньому змінювати контракт без порушення сумісності. Базовий набір ресурсів складається з колекцій users, access-levels, controllers, zones, logs, subscriptions, notifications та admin. Для кожного ресурсу визначено стандартний набір операцій CRUD: отримання списку та одного елемента, створення, оновлення, видалення. Додаткові операції необхідні для специфічних сценаріїв: /controllers/{id}/sync — примусова синхронізація, /subscriptions/{id}/invalidate — негайна блокада абонемента, /logs/bulk — масова передача подій від контролера.

Модель даних API відображає структуру реляційної бази. Об’єкт користувача (user) містить ідентифікатор, ПІБ, contact-дані, rfid_code, посилання на поточний subscription_id і access_level_id. Об’єкт subscription зберігає тип абонемента, дату

початку і закінчення дії, ліміти відвідувань. Контролер (controller) має унікальний device_id, ip_address, firmware_version, last_heartbeat і поле certificate_id або api_key для ідентифікації пристрою. Об’єкт log містить timestamp, user_id (якщо відомий), rfid_code (сировинний), zone_id, controller_id, event_type (access_granted / access_denied / error), та optional поле evidence (наприклад, знімок з камери або код помилки).

У межах протоколу обміну повідомленнями прийнято використовувати JSON як домінантний формат. Для контролерів пропонується контракт масової відправки подій: контролер агрегує події у масив і виконує один POST на /api/v1/logs/bulk зі списком подій у форматі [{controller_id, device_id, event_type, rfid_code, timestamp, zone_id, local_log_id}]. Сервер при успішному прийманні повертає список відповідностей local_log_id → global_log_id і статус для кожного запису. Ця масова операція дозволяє зменшити навантаження мережі та гарантувати ідемпотентну обробку через застосування заголовка Idempotency-Key. Для одиничної синхронної перевірки (коли контролер не має локальної копії прав або політика вимагає серверної верифікації) визначається endpoint POST /api/v1/authenticate з тілом {controller_id, device_id, rfid_code, timestamp, zone_id}; відповідь містить {result: "granted"/"denied", reason, expires_at}.

Механізм безпеки реалізується на декількох рівнях. Усі запити виконуються виключно через HTTPS/TLS. Для адміністраторських та користувацьких сесій застосовується JWT з невеликим терміном життя і механізмом оновлення через refresh token. Для пристроїв передбачено дві опції автентифікації: mutual TLS (рекомендовано для стаціонарних контролерів у корпоративній мережі) або HMAC-підпис у заголовку (X-Signature) для кожного запиту, де підпис обчислюється від тіла повідомлення й часу за секретним ключем, збереженим у пристрої та у сховищі секретів сервера. Крім того, необхідно впровадити ротацію ключів і механізми відкликання сертифікатів у разі компрометації пристрою.

Синхронізація між контролером і сервером має бути двосторонньою і стосуватися як прав доступу, так і журналів. Контролер періодично надсилає heartbeat на /api/v1/controllers/{id}/status, отримує у відповіді оновлення політик (наприклад, зміни в access-levels або список заблокованих rfid_code) і застосовує їх локально. У разі

відсутності зв’язку контролер працює автономно, застосовуючи локальну копію бланку дозволів, а після відновлення каналу виконує bulk upload локальних логів та отримує підтвердження синхронізації. Сервер, у свою чергу, зберігає часові мітки останньої синхронізації і контролює конфлікти по правилам «сервер перемагає» для політик, або по часу для логів (сервер зберігає першу отриману подію як джерело істини, але фіксує дублікати для аудиту).

Обробка помилок і винятків у API повинна бути уніфікованою: всі помилки повертаються у форматі JSON з полями error_code, message і, опціонально, details. Для контролерів особливо важливими є проміжні коди стану — наприклад, 202 Accepted для асинхронної обробки bulk-операцій, 409 Conflict при виявленні дубліката і 423 Locked при тимчасовому блокуванні ресурсу (наприклад, при спробі оновлення subscription, що модифікується іншим процесом). На рівні HTTP необхідно централізовано обробляти rate-limit і повертати відповідний Retry-After, щоб пристрої могли автоматично повторювати запит.

Аналітика та сповіщення організовуються як окремі сервіси, що підписані на події у системі. Після запису критичної події у лог (наприклад, множинні відмови доступу для одного rfid_code або невідповідність даних у картці клієнта) сервер ініціює генерацію нотифікації через Telegram API або Email SMTP і записує факт сповіщення у таблицю notifications. Для масштабу та відвантаження аналітичні задачі (побудова щоденних звітів, агрегування даних за зонами) виконуються асинхронно через чергу повідомлень (RabbitMQ або Kafka), що дозволяє не блокувати критичні операції автентифікації.

Підсумовуючи, проєктування REST API для системи контролю доступу спортивного комплексу вимагає комплексного підходу, що поєднує формалізацію ресурсів і сценаріїв, ретельне визначення контрактів обміну, багаторівневі механізми автентифікації та захисту, стратегії синхронізації пристроїв та серверної частини, а також продумані механізми моніторингу та масштабування. Розроблений варіант дозволяє забезпечити надійну й гнучку платформу, яка інтегрується з апаратною складовою і задовольняє вимоги безпеки, продуктивності та зручності адміністрування сучасного спортивного комплексу.

3 ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ СИСТЕМИ КОНТРОЛЮ ДОСТУПУ

3.1 Розробка серверної частини та мережевих сервісів

3.1.1 Налаштування сервера, роутінг та обробка запитів

У процесі реалізації комп’ютерної системи контролю доступу важливим етапом є створення серверної частини, яка забезпечує централізовану логіку обробки даних, взаємодію з базою даних та керування мережею пристроїв. Сервер виступає основним посередником між клієнтськими застосунками, контролерами доступу, зчитувачами карток та аналітичними сервісами. Його стабільна робота визначає цілісність і надійність усієї системи. У додатку Ж детально зображено структурну схему розподіленої комп'ютерної системи контролю доступу на базі мережі TCP/IP.

Для побудови серверної частини було обрано технологічний стек Node.js з фреймворком Express.js, що є оптимальним рішенням для розробки REST API-сервісів у реальному часі. Node.js забезпечує неблокуючу модель введення-виведення, що дозволяє ефективно обробляти велику кількість одночасних підключень, а Express.js надає гнучкий механізм маршрутизації (роутінгу) запитів і зручну інтеграцію з базами даних, системами логування та middleware-компонентами. Завдяки цим технологіям сервер здатний швидко реагувати на події, що надходять від контролерів і клієнтів, не створюючи затримок навіть за пікових навантажень.

Під час налаштування серверного середовища визначено основні елементи конфігурації: порт прослуховування запитів (наприклад, 8080), адреси підключення до бази даних, а також параметри безпеки — SSL-сертифікати, токен-автентифікацію та політику доступу (CORS). Сервер реалізовано як окремий модуль, який запускається у середовищі Node.js і функціонує в асинхронному режимі, що гарантує безперебійну роботу при великій кількості запитів.

Передусім виконується базове налаштування сервера. На цьому етапі створюється структура директорій проєкту, встановлюються необхідні бібліотеки та залежності (express, cors, body-parser, dotenv тощо). У конфігураційному файлі .env задаються параметри середовища — номер порту, адреса бази даних, ключі доступу.

Після ініціалізації середовища створюється головний файл сервера, наприклад server.js, у якому виконується налаштування базових маршрутів, підключення проміжних обробників (middleware) і запуск прослуховування порту. Нижче наведено код базової конфігурації сервера.

Лістинг 3.1 — Базова конфігурація сервера

```
const express = require('express');
const app = express();
const cors = require('cors');
app.use(cors());
app.use(express.json());
// Підключення маршрутів
const usersRouter = require('./routes/users');
const zonesRouter = require('./routes/zones');
app.use('/api/users', usersRouter);
app.use('/api/zones', zonesRouter);
// Запуск сервера
const PORT = process.env.PORT || 3000;
app.listen(PORT, () => {
  console.log(`Сервер запущено на порту ${PORT}`);
});
```

У системі передбачено чітке розділення логіки за маршрутами. Наприклад, усі запити, пов’язані з користувачами, обробляються у файлі users.js, а маршрути для зон доступу — у файлі zones.js. Такий підхід відповідає архітектурі MVC (Model-View-Controller), де контролери відповідають за обробку запитів і взаємодію з моделями бази даних.

Система маршрутизації (роутінгу) реалізує логіку обробки HTTP-запитів, які надходять до серверу. Кожен запит спрямовується до відповідного контролера, що відповідає за певну функціональність системи. Наприклад:

— /api/v1/users маршрути для роботи з користувачами (створення, редагування, видалення, отримання списку);

— /api/v1/controllers маршрути для взаємодії з апаратними контролерами доступу;

— /api/v1/zones управління зонами проходу;

— /api/v1/logs запис і перегляд подій у журналі системи.

Кожен маршрут реалізує набір стандартних операцій REST: GET — отримання даних, POST — створення нового запису, PUT/PATCH — оновлення існуючих даних та DELETE — видалення ресурсу.

Такий підхід дозволяє дотримуватися принципів RESTful-архітектури та спрощує інтеграцію системи з іншими клієнтами або сервісами. Для обробки запитів використовується набір middleware-компонентів, які виконують послідовну перевірку й підготовку даних. Зокрема:

— authMiddleware перевірка токена авторизації користувача;

— validateRequest перевірка коректності структури запиту (наприклад, формат JSON, обов’язкові поля);

— errorHandler централізована обробка помилок із поверненням стандартного HTTP-коду відповіді;

— logger реєстрація усіх запитів та відповідей у журналі серверних подій.

Таке багаторівневе опрацювання забезпечує високу надійність і безпечність системи, дозволяючи виявляти помилки ще до того, як вони досягнуть бізнес-логіки.

Приклад реалізації маршруту для отримання списку користувачів наведено нижче:

Лістинг 3.2 — Реалізація маршруту для отримання списку користувачів

```
const express = require('express');
const router = express.Router();
const db = require('../database');
router.get('/', async (req, res) => {
  try {
    const users = await db.query('SELECT * FROM Users');
    res.status(200).json(users.rows);
  } catch (error) {
    console.error('Помилка при отриманні користувачів:', error);
    res.status(500).json({ message: 'Внутрішня помилка сервера' });
  }
});
module.exports = router;
```

У структурі коду кожен контролер реалізує власні методи взаємодії з базою даних через шар data access layer (DAL) або ORM (Object-Relational Mapping). Це дає можливість абстрагуватися від конкретної СУБД і робить систему легко переносною на інші платформи. Наприклад, у контролері користувачів метод createUser() створює новий запис у базі після перевірки валідності даних, а метод getUserById() виконує пошук за унікальним ідентифікатором користувача.

Обробка запитів від контролерів доступу відбувається в окремому модулі, який працює у режимі реального часу. Коли апаратний пристрій надсилає дані про спробу входу (ID картки, час, статус), сервер приймає запит, перевіряє права доступу користувача в базі та формує відповідь — «доступ дозволено» або «відмовлено». Водночас подія автоматично записується в журнал (Logs), що дозволяє здійснювати аудит усіх дій у системі.

Додатково сервер реалізує механізм вебсокет-з’єднання (WebSocket) для передачі даних у реальному часі. Це дозволяє адміністративній панелі миттєво отримувати оновлення — наприклад, повідомлення про відкриття дверей або помилку при зчитуванні картки. Такий підхід забезпечує високий рівень інтерактивності системи й дозволяє адміністраторам оперативно реагувати на події.

Після налаштування всіх компонентів сервер проходить тестування на коректність обробки запитів, стабільність роботи під навантаженням і безпечність передачі даних. Результати перевірки демонструють, що обрана архітектура забезпечує низьку затримку відповідей, масштабованість і можливість подальшого розширення функціоналу без зміни базової логіки.

Для перевірки коректності роботи REST API було розгорнуто тестовий сервер на платформі Node.js із використанням фреймворку Express.js. Після запуску серверної частини за адресою *http://localhost:3000* здійснювалось звернення до маршруту */api/users* безпосередньо через веб-браузер. У результаті сервер повернув коректно сформований JSON-об’єкт, що містить перелік зареєстрованих користувачів із відповідними ідентифікаторами та ролями.

На рисунку 3.1 наведено приклад успішного запиту, який демонструє працездатність маршруту та правильність обробки даних. Такий підхід дає змогу

переконатися у функціональності REST API без використання сторонніх інструментів, оскільки браузер може виконувати стандартні GET-запити до сервера.

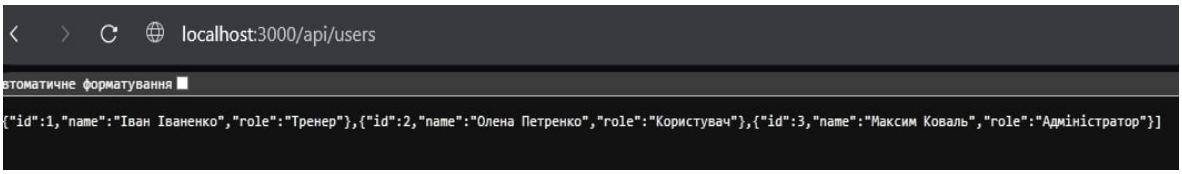

Рисунок 3.1 — Перевірка роботи REST API

Під час тестування також було перевірено стабільність роботи сервера при повторних запитах, коректність обробки JSON-відповідей і відсутність критичних помилок у логах. Це підтвердило, що розроблена структура маршрутизації й обробки запитів функціонує відповідно до вимог архітектури REST та може бути розширена для підтримки інших методів, таких як POST, PUT або DELETE.

Особливу увагу надалі планується приділити реалізації механізмів автентифікації користувачів за допомогою токенів доступу (JWT) та middleware-функцій, які забезпечують перевірку дійсності сесій. Це дозволить обмежити доступ до адміністративних маршрутів і гарантувати безпечну взаємодію клієнтів із сервером.

Таким чином, розроблена серверна частина забезпечує ефективний обмін інформацією між усіма компонентами системи контролю доступу. Використання REST API гарантує гнучкість інтеграції з іншими підсистемами, а застосування Node.js та Express забезпечує стабільну роботу, масштабованість і високу продуктивність системи.

3.1.2 Модуль авторизації та управління ролями користувачів

Модуль авторизації є ключовим компонентом системи контролю доступу, оскільки саме він забезпечує перевірку особи користувача та визначення його прав у межах системи. Основне завдання цього модуля — гарантувати, що доступ до певних ресурсів або функцій отримують лише ті користувачі, які мають відповідний рівень прав.

Процес авторизації починається з аутентифікації, тобто перевірки правильності введених користувачем облікових даних. При вході користувач вводить логін і пароль,

які надсилаються на сервер через захищений запит (наприклад, методом POST до маршруту /api/login). Сервер перевіряє дані, порівнюючи введений пароль із хешованим значенням, що зберігається в базі даних. Для хешування використовується алгоритм bcrypt, який забезпечує криптографічний захист паролів навіть у випадку витоку бази даних [25].

У разі успішної аутентифікації сервер генерує токен доступу (JWT — JSON Web Token), який містить інформацію про користувача, його роль та термін дії токена. Цей токен надсилається клієнту й зберігається на його боці (зазвичай у браузері або мобільному додатку). При кожному подальшому запиті до серверу клієнт додає токен до заголовка запиту. Сервер перевіряє дійсність токена за допомогою middleware-функції. Якщо токен є справжнім і не простроченим, користувач отримує доступ до запитуваного ресурсу.

Для реалізації управління ролями в системі створено таблицю AccessLevels, де визначено типи ролей. Кожному користувачу таблиці Users присвоюється певний рівень доступу через зовнішній ключ access_level_id. Таким чином реалізується зв’язок типу “один до багатьох”: один рівень доступу може належати кільком користувачам, але кожен користувач має лише один рівень доступу.

Адміністратор системи має найвищі права: може створювати нових користувачів, редагувати їх ролі та відстежувати активність у журналі (Logs). Звичайні користувачі мають доступ лише до власних даних або обмежених функцій. Такий підхід забезпечує принцип role-based access control (RBAC) — розмежування прав доступу відповідно до ролі користувача.

Крім того, модуль передбачає перевірку дозволів на рівні кожного запиту. Якщо користувач із нижчим рівнем доступу намагається виконати адміністративну дію, сервер повертає помилку *403 Forbidden*. Це унеможливлює несанкціонований вплив на критичні дані або конфігурацію системи.

Таким чином, розроблений модуль авторизації та управління ролями користувачів забезпечує:

— безпечну перевірку користувачів при вході;

— надійне зберігання паролів за допомогою хешування;

— генерацію та валідацію токенів доступу (JWT);

— гнучке управління правами через систему ролей;

— захист адміністративних маршрутів від стороннього доступу.

3.1.3 Сервіс обробки запитів від контролерів доступу

Сервіс обробки запитів від контролерів доступу є центральним елементом системи, що забезпечує взаємодію між фізичними пристроями контролю (зчитувачами, сенсорами, контролерами) та серверною частиною програмного забезпечення. Його основна функція полягає у прийманні, обробці та збереженні даних, що надходять від апаратних компонентів, а також у передачі відповідей або команд у зворотному напрямку — до контролерів.

Робота сервісу побудована на принципах клієнт-серверної архітектури. Контролери виступають клієнтами, які періодично або подієво надсилають запити до центрального сервера. Такі запити можуть містити інформацію про події доступу (наприклад, “зчитано картку”, “відмовлено у доступі”, “відкрито двері”), або запитувати інструкції від сервера (“чи дозволено вхід користувачу з таким ідентифікатором?”). Сервер, у свою чергу, обробляє ці запити, перевіряє відповідні записи у базі даних (користувачів, ролей, зон тощо) і формує відповідь у вигляді структурованого повідомлення (JSON).

Для реалізації обміну даними використовується REST API, що дозволяє контролерам надсилати запити до певних маршрутів, таких як:

— POST /api/access/check перевірка дозволу на доступ;

— POST /api/access/log надсилання інформації про подію доступу;

— GET /api/controllers/status отримання поточного стану контролера.

Комунікація між пристроями і сервером здійснюється через протокол HTTP/HTTPS, що забезпечує сумісність із більшістю мережевих пристроїв і можливість віддаленого керування. Для критичних даних передбачене використання HTTPS-з’єднання, яке гарантує шифрування переданої інформації.

На серверному боці сервіс реалізує логіку приймання запиту, розбору даних, перевірки автентичності пристрою (через унікальний токен або сертифікат), після чого передає інформацію у відповідний обробник. У разі позитивної перевірки сервер може надати дозвіл на доступ, відкрити замок або записати подію у таблицю Logs, зазначивши користувача, час та тип дії.

Важливою частиною цього модуля є система черг і подій, що дозволяє обробляти кілька запитів одночасно без перевантаження сервера. Використовується асинхронна обробка (через механізм промісів і callback-функцій у Node.js), що забезпечує високу продуктивність і стабільність роботи системи навіть при великій кількості підключених контролерів.

Окрему увагу приділено захисту комунікацій між сервером і контролерами. Для цього передбачено перевірку автентичності пристроїв перед прийманням запитів, а також обмеження доступу за IP-адресами або ключами API. Усі запити фіксуються у журналі (Logs) із зазначенням часу, типу операції та результату обробки, що дозволяє здійснювати аудит роботи системи.

3.2 Розробка клієнтської частини (веб-інтерфейсу) для адміністратора

У системі контролю доступу веб-інтерфейс адміністратора виконує ключову роль, оскільки саме через нього здійснюється управління всіма аспектами функціонування системи — від контролю користувачів і пристроїв до аналізу статистики та моніторингу подій у реальному часі. Розробка клієнтської частини є одним із найбільш важливих етапів, оскільки від її зручності, наочності та стабільності залежить ефективність роботи персоналу спортивного комплексу.

Веб-інтерфейс створюється з використанням сучасних вебтехнологій — HTML5, CSS3, JavaScript та фреймворку React.js, який дозволяє реалізувати компонентний підхід до побудови інтерфейсу. Такий підхід забезпечує високу швидкодію, модульність і простоту масштабування додатку. Завдяки використанню бібліотек для адаптивного дизайну (наприклад, Bootstrap або Tailwind CSS), веб-інтерфейс коректно відображається на різних пристроях — від комп’ютера адміністратора до планшета або ноутбука, що дає змогу використовувати систему у будь-якому робочому середовищі.

Основною метою розробки клієнтської частини є забезпечення зручного управління користувачами, зонами доступу та контролерами. На головній панелі адміністратора розроблено розділи:

— «Користувачі» (Users) відображає перелік усіх зареєстрованих користувачів системи із можливістю додавання, редагування та видалення записів. Також реалізовано функції пошуку за прізвищем, ID або рівнем доступу.

— «Контролери» (Controllers) надає адміністратору можливість переглядати стан підключених пристроїв, оновлювати їхнє програмне забезпечення та синхронізувати налаштування з сервером.

— «Зони» (Zones) використовується для управління логічними зонами комплексу, у яких діють різні права доступу.

— «Журнал подій» (Logs) містить хронологічний запис усіх подій: входи, спроби несанкціонованого доступу, збої у з’єднанні та адміністративні дії.

Кожна сторінка взаємодіє із сервером через REST API, який було спроектовано на попередніх етапах. Наприклад, при відкритті розділу “Користувачі” клієнтський додаток надсилає запит GET /api/users, отримує список користувачів у форматі JSON і відображає його у вигляді інтерактивної таблиці. Для додавання нового користувача використовується метод POST, а для оновлення чи видалення — відповідно PUT або DELETE. Таким чином забезпечується повна інтеграція між клієнтською та серверною частинами системи.

Особлива увага приділяється інформаційній безпеці веб-інтерфейсу. Передбачено механізм авторизації адміністратора за допомогою JWT-токена, який зберігається у локальному сховищі браузера (localStorage). Це дає змогу гарантувати, що доступ до адміністративної панелі мають лише користувачі з відповідними повноваженнями. У випадку спроби доступу без авторизації система автоматично перенаправляє користувача на сторінку входу.

Для підвищення зручності адміністрування передбачено інтерактивну карту комплексу, реалізовану з використанням бібліотеки Leaflet.js. На ній позначені всі зони та пристрої контролю доступу. Адміністратор може в реальному часі бачити стан

кожного пристрою (активний, неактивний, з помилкою), а також натисканням на позначку отримати детальну інформацію або виконати керуючу дію (перезапуск, оновлення конфігурації тощо). Такий підхід значно спрощує моніторинг системи та візуалізує її роботу.

Додатково реалізовано модуль сповіщень, який дозволяє миттєво повідомляти адміністратора про критичні події, наприклад, про збої у зв'язку з контролером або несанкціоновану спробу входу. Повідомлення відображаються у вигляді спливаючих вікон у браузері, а також можуть дублюватися через Telegram-бот або email, що розширює можливості оперативного реагування.

З технічного погляду клієнтська частина реалізована як односторінковий додаток (SPA — Single Page Application), що забезпечує швидке перемикання між розділами без повторного завантаження сторінки. Обмін даними із сервером відбувається асинхронно, що значно підвищує продуктивність і зручність користування. При цьому структура коду побудована таким чином, щоб її можна було легко розширювати у майбутньому — наприклад, додати функції управління абонементами або статистичну аналітику відвідувань.

Для зручності супроводу та розширення функціоналу клієнтської частини веб-застосунок структуровано за модульним принципом. Усі вихідні файли розміщено в каталозі /src, який містить кілька логічно розділених підкаталогів, що відповідають окремим аспектам функціонування програми.

На рисунку 3.2 представлена структура файлів та каталогів клієнтської частини веб-додатку, розробленого для адміністрування системи контролю доступу. Ця архітектура базується на модульному підході (з використанням бібліотеки React, про що свідчать розширення .jsx), що забезпечує високу масштабованість, зручність підтримки та незалежність окремих функціональних блоків.

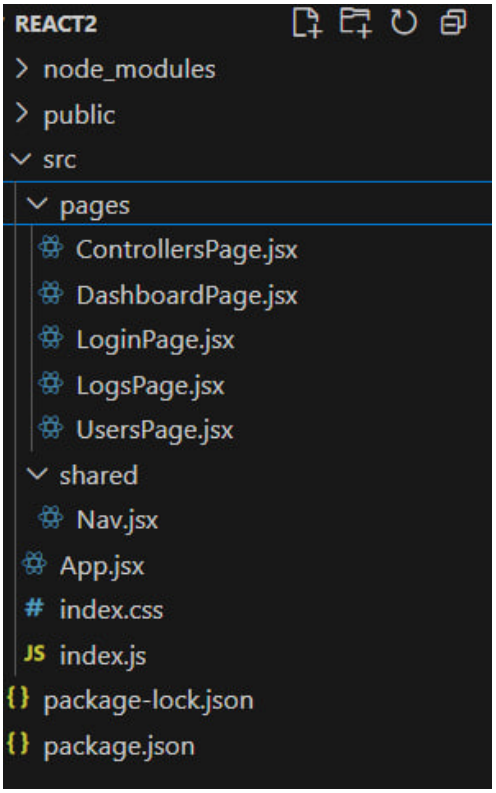

Рисунок 3.2 — Структура файлів проекту

Основним елементом є каталог src/pages, який містить самодостатні компоненти, що відповідають за ключові екрани веб-інтерфейсу адміністратора:

— LoginPage.jsx відповідає за функціонал автентифікації, забезпечуючи захист системи від несанкціонованого доступу;

— DashboardPage.jsx центральний модуль візуалізації, призначений для відображення ключових показників (KPI) та зведеної аналітики активності комплексу;

— UsersPage.jsx реалізує повний цикл управління користувачами (CRUD-операції: створення, перегляд, оновлення, видалення), що є необхідним для адміністрування облікових записів та абонементів;

— ControllersPage.jsx виділений модуль для моніторингу та конфігурації апаратних пристроїв — контролерів доступу, що забезпечує централізоване управління СКУД;

— LogsPage.jsx призначений для відображення та фільтрації журналу подій у реальному часі, що критично важливо для оперативного контролю та розслідування інцидентів.

Основний файл App.js виконує роль маршрутизатора і визначає логіку доступу до сторінок: якщо користувач авторизований (дані про сесію збережено у localStorage), йому відкривається головна панель; у протилежному випадку — сторінка входу. Файл index.js ініціалізує застосунок, підключає глобальні стилі та монтує головний компонент у DOM-дерево браузера.

Така чітка організація структури проєкту демонструє відповідність розробки принципам функціональної декомпозиції та забезпечує надійну основу для подальшого розширення системи, включно з додаванням нових аналітичних інструментів та інтеграцією додаткових мережевих сервісів, як це передбачено метою дослідження. Блок-схема алгоритму роботи клієнтської частини зображено у додатку И.

Розроблений веб-інтерфейс адміністратора реалізує всі ключові функції системи контролю доступу та забезпечує інтуїтивну взаємодію користувача з даними. Через панель адміністратор може переглядати список зареєстрованих користувачів, створювати нові облікові записи або редагувати наявні, призначати рівні доступу, контролювати роботу пристроїв (контролерів) і аналізувати журнал подій у реальному часі. Додатково інтерфейс підтримує пошук, фільтрацію та сортування записів, що суттєво спрощує адміністрування при великій кількості користувачів і зон. У майбутньому така архітектура дозволяє без складних змін додати модулі аналітики, візуалізацію активності на інтерактивній карті або автоматизоване сповіщення про інциденти. Завдяки цьому клієнтська частина стає не просто засобом керування, а повноцінним аналітичним інструментом, який інтегровано у загальну інфраструктуру безпеки спортивного комплексу.

Таким чином, клієнтська частина системи контролю доступу виступає не просто інтерфейсом взаємодії користувача з системою, а повноцінним інструментом управління, який поєднує функції контролю, моніторингу, аналітики та безпеки. Завдяки використанню сучасних вебтехнологій і продуманій архітектурі вона

забезпечує ефективну роботу персоналу спортивного комплексу, мінімізує кількість помилок та підвищує рівень автоматизації процесів.

3.3 Розробка функціоналу для забезпечення контролю доступу

Функціонал контролю доступу є центральним елементом розробленої системи, оскільки саме він забезпечує реалізацію основного призначення — регулювання, моніторинг та фіксацію входів і виходів користувачів у межах спортивного комплексу. На цьому етапі здійснюється інтеграція між апаратним рівнем (контролери, зчитувачі, замки) та програмним забезпеченням, що відповідає за прийняття рішень і управління правами доступу.

Розроблений функціонал передбачає багаторівневу взаємодію компонентів системи. На фізичному рівні контроль здійснюється за допомогою RFID-зчитувачів, які розпізнають ідентифікатори користувачів, та контролерів InBio, що приймають рішення про відкриття або блокування дверей. На логічному рівні працює серверна частина, яка через REST API обробляє отримані запити, перевіряє їх у базі даних і повертає відповідь контролеру. Якщо користувач має чинний дозвіл на вхід у певну зону — система подає сигнал на електромагнітний замок, а подія записується у журнал подій (Logs).

Однією з основних задач функціоналу є динамічне управління правами доступу. Адміністратор через веб-інтерфейс має можливість оперативно змінювати рівні доступу для окремих користувачів або груп. Наприклад, у спортивному комплексі це можуть бути різні категорії — відвідувачі, тренери, персонал обслуговування, керівництво. Кожній категорії надається доступ лише до певних зон (тренажерний зал, адміністративний корпус, склад інвентарю тощо). Усі зміни автоматично синхронізуються між клієнтською частиною, сервером та контролерами, що забезпечує актуальність даних у реальному часі.

Важливою складовою розробленого функціоналу є механізм перевірки дійсності ідентифікаторів. Система підтримує кілька типів носіїв — безконтактні карти, брелоки або мобільні ідентифікатори (через NFC). Кожен із них має унікальний код, який зв’язується з конкретним обліковим записом користувача в базі даних. При кожній

спробі доступу контролер звіряє отриманий код із записами в системі, перевіряє термін дії пропуску та відповідність дозволам. У разі виявлення несанкціонованої спроби вхід блокується, а адміністратор отримує сповіщення через веб-інтерфейс.

Розроблена система також реалізує журнальний контроль усіх подій. Кожен факт входу, відмови у доступі, зміни прав користувачів або оновлення контролера фіксується у таблиці Logs, що дозволяє проводити аудит і аналіз дій користувачів. Це особливо важливо для спортивних комплексів, де може бути необхідно простежити відвідуваність, виявити несанкціоновані входи або перевірити дисципліну персоналу. Завдяки модулю фільтрації та пошуку подій адміністратор може легко переглянути всі дії за певний період або конкретного користувача.

Окрім базових механізмів, у систему закладено інтелектуальні елементи контролю. Наприклад, при багаторазових невдалих спробах входу користувач автоматично блокується на визначений проміжок часу, а подія реєструється як потенційна загроза безпеці. Також передбачено можливість налаштування графіків доступу, що дозволяє обмежувати час відвідування певних зон (наприклад, доступ тренерів лише у робочі години). Це забезпечує додатковий рівень безпеки та гнучкість у керуванні ресурсами комплексу.

Функціонал контролю доступу розроблено з урахуванням масштабованості. Система підтримує підключення додаткових контролерів, зчитувачів або зон без необхідності змін у базовому коді. Усі нові пристрої автоматично реєструються в системі після первинної ініціалізації, а адміністратор може призначити для них параметри роботи через веб-панель. Такий підхід робить систему придатною як для невеликих спортивних клубів, так і для великих спортивних комплексів із десятками зон контролю.

Логіка обробки подій контролю доступу реалізована у вигляді взаємодії між кількома компонентами системи — контролером, сервером застосунку, базою даних та веб-інтерфейсом адміністратора. Основна мета цієї логіки полягає у забезпеченні швидкої та коректної перевірки прав користувача, фіксації події та передачі відповідного сигналу на пристрій управління доступом.

Процес обробки події проходить кілька послідовних етапів. Ідентифікація користувача – при прикладанні RFID-карти або мобільного ідентифікатора до зчитувача, контролер отримує унікальний код користувача (UID) (рисунок 3.3).

Цей код передається на сервер через локальну мережу у вигляді запиту HTTP або MQTT (залежно від реалізації мережевого протоколу). Другий етап це перевірка даних у базі. Серверна частина приймає запит і звертається до бази даних для перевірки відповідності UID запису користувача. Якщо запис знайдено, система перевіряє статус користувача (активний/заблокований), термін дії абонементу або пропуску, а також список дозволених зон для цього користувача. Потім йде прийняття рішення.

На основі отриманих даних формується рішення про надання або відмову у доступі. Якщо користувач має право входу у дану зону — сервер надсилає контролеру команду *access_granted*, після чого подається сигнал на розблокування замка. У протилежному випадку передається команда *access_denied*, що супроводжується світловою або звуковою індикацією на пристрої.

Незалежно від результату, кожна подія фіксується у таблиці *Logs* з такими параметрами: час події, UID користувача, ідентифікатор контролера (зона доступу), результат події (доступ дозволено / відмовлено), причина відмови. Така фіксація забезпечує можливість подальшого аудиту та аналітики.

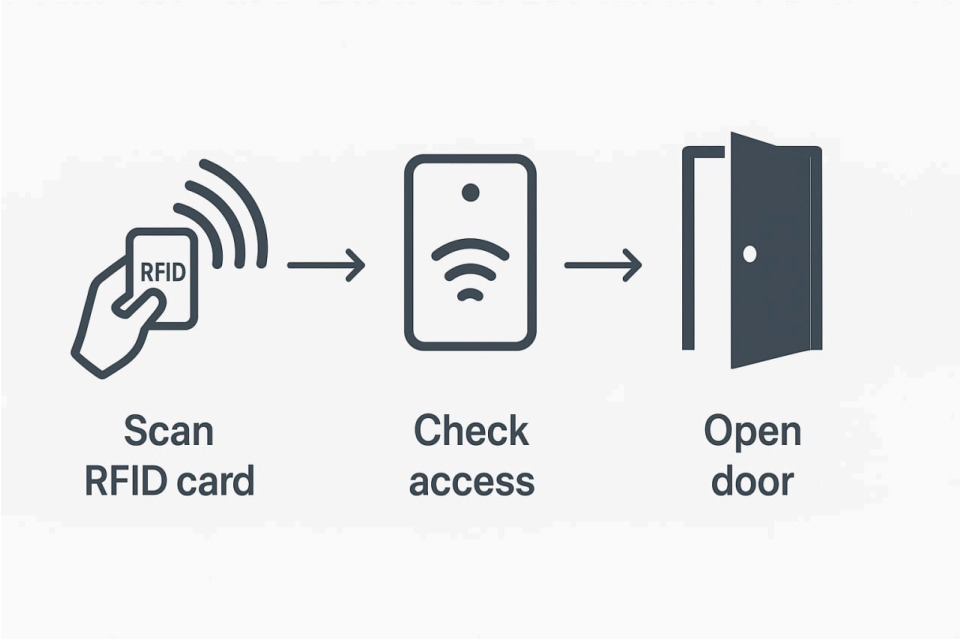

Рисунок 3.3 — Схема процесу контролю доступу

Для підвищення продуктивності система використовує кешування дозволів у пам’яті сервера. Це дозволяє уникати частих запитів до бази даних при масових зверненнях, наприклад, у години пік, коли багато користувачів одночасно проходять через турнікет.

Архітектурно логіка контролю доступу побудована за принципом "клієнт–сервер–контролер", що забезпечує відмовостійкість: у випадку тимчасової втрати зв’язку з сервером контролер зберігає у локальній пам’яті останній список дозволених UID і продовжує роботу в автономному режимі. Після відновлення з’єднання дані автоматично синхронізуються.

Взаємодія між серверною та клієнтською частинами системи контролю доступу реалізована за принципами REST-архітектури (Representational State Transfer), що забезпечує простий, масштабований і стандартизований спосіб обміну даними між компонентами (рисунок 3.4).

Комунікація здійснюється через HTTP-запити, у яких клієнтська частина (веб-інтерфейс адміністратора) звертається до серверних маршрутів для отримання або оновлення інформації.

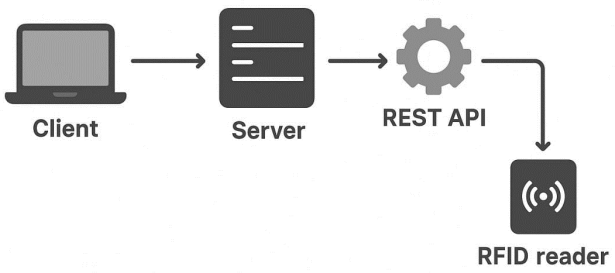

Рисунок 3.4 — Схема взаємодії клієнтської та серверної частин системи контролю доступу

Передача даних відбувається у форматі JSON, що забезпечує легке серіалізування та сумісність із більшістю сучасних мов програмування. Наприклад,

при завантаженні сторінки «Користувачі» клієнт надсилає запит GET /api/users, у відповідь сервер повертає масив об’єктів користувачів із зазначенням їхніх ролей, зон і статусу доступу. Аналогічно, при створенні нового користувача відправляється запит POST /api/users із JSON-даними форми, а при редагуванні — PUT /api/users/:id.

Для підвищення безпеки система реалізує автентифікацію за токеном. Після успішного входу користувача сервер генерує унікальний JWT-токен (JSON Web Token), який зберігається у локальному сховищі браузера (localStorage). Усі подальші запити до захищених маршрутів супроводжуються цим токеном у заголовку Authorization. На стороні сервера токен перевіряється middleware-функцією, що дозволяє отримати інформацію про поточного користувача та визначити, чи має він необхідні права доступу.

Також у системі передбачено механізм оновлення даних у реальному часі. Зокрема, події, що надходять від контролерів (наприклад, спроби входу або виходу), передаються до сервера через API-маршрут /api/logs, а далі надсилаються до клієнтської частини за допомогою WebSocket-з’єднання або періодичних запитів. Завдяки цьому журнал подій оновлюється динамічно без необхідності перезавантаження сторінки.

Усі передані дані проходять попередню валідацію — як на клієнтському, так і на серверному рівні. На клієнті перевіряється правильність заповнення форм, а на сервері — цілісність запиту та відповідність даних визначеній структурі бази. У разі помилки або порушення прав доступу система повертає повідомлення у стандартизованому форматі, що спрощує обробку відповідей і підвищує стабільність роботи всієї системи.

Загалом, розроблений функціонал є гнучким, безпечним та ефективним рішенням для управління доступом у спортивних закладах. Його реалізація забезпечує повну інтеграцію апаратного та програмного рівнів, оперативне адміністрування, захист від несанкціонованих дій і можливість розширення функціоналу без значних витрат.

3.4 Реалізація статистичної та аналітичної підсистеми

У межах розробленої системи контролю доступу важливе місце посідає підсистема збору, обробки та аналізу статистичних даних, яка забезпечує можливість не лише контролю поточних подій, але й глибокого розуміння тенденцій використання інфраструктури спортивного комплексу. Її основне призначення полягає у відстеженні активності користувачів, контролерів та зон доступу з метою підвищення ефективності управління об’єктом, оптимізації ресурсів і забезпечення належного рівня безпеки.

Аналітична підсистема функціонує як окремий логічний модуль серверної частини, що взаємодіє з базою даних та API системи. Вона періодично або у режимі реального часу здійснює збір інформації з журналів подій, записів користувачів, логів контролерів, історії спроб доступу, а також інших супровідних даних. Усі отримані записи зберігаються в централізованій базі даних, що дозволяє гарантувати цілісність, узгодженість і доступність інформації для подальшого аналізу.

Для аналітичної обробки застосовуються методи агрегації, фільтрації, сортування та статистичного підрахунку. Наприклад, на основі SQL-запитів або спеціальних функцій REST API можна отримати узагальнені відомості про кількість входів та виходів за обраний період, визначити пікові години навантаження, а також зафіксувати випадки несанкціонованих спроб доступу. Результати можуть бути представлені у форматі JSON, що забезпечує простоту інтеграції з клієнтською частиною системи.

У веб-інтерфейсі адміністратора реалізовано модуль відображення аналітичних показників, який надає можливість переглядати зведення у вигляді діаграм, графіків та таблиць (рисунок 3.5). Серед основних параметрів, що можуть бути відображені:

— кількість успішних та невдалих спроб входу за певний період;

— розподіл активності користувачів по зонах;

— середня тривалість перебування в приміщеннях;

— пікові години відвідуваності;

— відсоток збоїв або технічних помилок у роботі контролерів.

Отримані дані можуть бути використані не лише для оперативного моніторингу, але й для стратегічного планування розвитку інфраструктури. На основі аналітичних

звітів адміністрація спортивного комплексу може приймати обґрунтовані управлінські рішення — наприклад, щодо оптимізації графіків роботи персоналу, перерозподілу зон доступу, або вдосконалення політики безпеки.

Рисунок 3.5 — Графік відображення аналітичних показників

Таким чином, аналітична підсистема виступає інтелектуальним ядром системи контролю доступу. Вона перетворює накопичені дані на цінну управлінську інформацію, сприяючи підвищенню ефективності функціонування комплексу та забезпечуючи можливість прогнозування потенційних ризиків і проблем у майбутньому. Це робить систему не лише інструментом контролю, а й потужним засобом для прийняття рішень на основі даних.

4 ТЕСТУВАННЯ ТА ВПРОВАДЖЕННЯ КОМП'ЮТЕРНОЇ СИСТЕМИ КОНТРОЛЮ ДОСТУПУ

4.1 Розробка плану та методик тестування

Після завершення розробки серверної, клієнтської та апаратної частин системи контролю доступу спортивного комплексу було проведено комплексне тестування, метою якого є перевірка коректності, стабільності, безпеки та зручності роботи всіх компонентів.

Тестування дозволяє переконатися, що система відповідає вимогам технічного завдання, а також гарантує надійність її функціонування в реальних умовах експлуатації.

План тестування базувався на таких принципах:

— поступовість — перевірка від окремих модулів до інтегрованої системи;

— повнота покриття — охоплення всіх сценаріїв, які можуть виникати під час роботи комплексу;

— реалістичність умов — моделювання ситуацій, максимально наближених до фактичного користування (зчитування карток, блокування, спроби несанкціонованого входу);

— документування — фіксація всіх результатів тестів для подальшого аналізу.

Після завершення етапу розробки програмних модулів системи контролю доступу було проведено комплексне тестування, спрямоване на перевірку правильності, надійності та зручності функціонування всіх складових системи. Основна мета тестування полягала у виявленні можливих помилок, перевірці відповідності системи вимогам технічного завдання, а також оцінці її готовності до експлуатації у спортивному комплексі.

Тестування здійснювалося поетапно — від перевірки окремих компонентів до перевірки всієї системи в цілому. На першому етапі було проведено модульне тестування, під час якого перевірялась робота кожного програмного елемента окремо. Зокрема, тестувалися функції серверної частини, які відповідають за обробку запитів REST API, збереження та отримання даних із бази, а також механізми авторизації

користувачів. Це дозволило виявити локальні помилки у логіці обробки даних та виправити їх ще до об’єднання всіх модулів.

Після цього було проведено інтеграційне тестування, метою якого стала перевірка взаємодії між сервером, базою даних, веб-інтерфейсом адміністратора та імітованими контролерами доступу. На цьому етапі перевірялося, чи правильно передаються запити між компонентами, чи відображаються зміни у базі даних після дій користувача, і чи надходять відповідні події від контролерів у журнал. Іншими словами, тестувалася узгодженість роботи програмних частин системи між собою.

Наступним етапом стало системне тестування, у межах якого оцінювалась робота комплексу як єдиного цілого. Було перевірено повний цикл дії — від моменту прикладання картки користувача до зчитувача і передачі сигналу на сервер до запису події у журнал і відображення її у веб-інтерфейсі адміністратора. Також моделювалися різні сценарії: спроба входу користувача без відповідного рівня доступу, тимчасова відсутність з’єднання між контролером і сервером, блокування користувача в базі. Це дозволило переконатися, що система стабільно працює навіть у нештатних умовах.

Після цього було виконано функціональне тестування, яке мало на меті перевірити, чи реалізовано всі заявлені функції згідно з вимогами. Тестувались можливості створення користувачів, призначення ролей, налаштування зон доступу, реєстрація подій, ведення журналів і формування статистичних звітів. За результатами цього етапу підтверджено, що система виконує всі основні функції без збоїв, а дані відображаються коректно у всіх модулях.

Завершальним етапом стало тестування безпеки, яке охоплювало перевірку системи авторизації, обмеження доступу до адміністративних розділів, коректність обробки токенів авторизації (JWT), а також захист від несанкціонованих запитів. Під час перевірки імітувались спроби входу без дійсних облікових даних, SQL-ін’єкції та втручання у передані параметри. Система успішно заблокувала всі спроби несанкціонованого доступу, що підтвердило її надійність.

Таким чином, проведене поетапне тестування дозволило переконатися у працездатності системи контролю доступу, її стабільності та безпечності. Усі компоненти функціонують злагоджено, забезпечуючи своєчасну обробку подій,

коректне відображення інформації в інтерфейсі адміністратора та захист даних користувачів. Результати тестування підтверджують готовність розробленого програмного комплексу до впровадження в реальні умови експлуатації спортивного комплексу.

4.2 Функціональне тестування веб-інтерфейсу

Тестування веб-застосунку є одним із ключових етапів розробки, оскільки дозволяє перевірити, наскільки система відповідає функціональним вимогам, забезпечує стабільну роботу всіх елементів інтерфейсу та коректно взаємодіє з базою даних. Для тестування застосунку admin-dashboard спершу налаштовується робоче середовище. У директорії проекту виконуються такі команди:

— npm install встановлення всіх необхідних залежностей та бібліотек, що забезпечують роботу клієнтської та серверної частини;

— npm run watch — автоматичне оновлення компонентів веб-застосунку при зміні коду, що дозволяє відразу бачити внесені зміни у браузері;

— npm start — запуск клієнтської частини веб-додатку;

— npm run start — запуск вебсерверу для обробки запитів користувачів.

Після цього проведемо функціональне тестування веб-інтерфейсу, яке передбачає перевірку роботи всіх кнопок, форм, меню, таблиць, графіків та інших елементів панелі адміністратора. Основною метою цього тестування є впевненість у тому, що всі функції вебзастосунку працюють правильно, а дані з бази коректно відображаються та змінюються.

Функціональне тестування веб-інтерфейсу передбачає перевірку коректності роботи елементів адміністративної панелі та взаємодії з базою даних. Кожна кнопка, форма чи таблиця проходить перевірку на відповідність вимогам системи.

Першим етапом тестування є перевірка сторінки авторизації. Адміністратор повинен ввести логін та пароль для доступу до панелі управління (рисунок 4.1).

Під час тестування перевіряється, що при введенні правильних облікових даних адміністратор потрапляє на головну панель без помилок. Якщо ж вводяться неправильний пароль або логін, система негайно інформує про помилку, не дозволяючи доступу (рисунок 4.2). У випадку, коли користувач залишає поля порожніми, вебзастосунок підказує, що ці поля обов'язкові для заповнення. Також перевіряється, як система реагує на введення спеціальних символів чи надзвичайно довгих рядків, щоб гарантувати стійкість до некоректного вводу або потенційних атак.

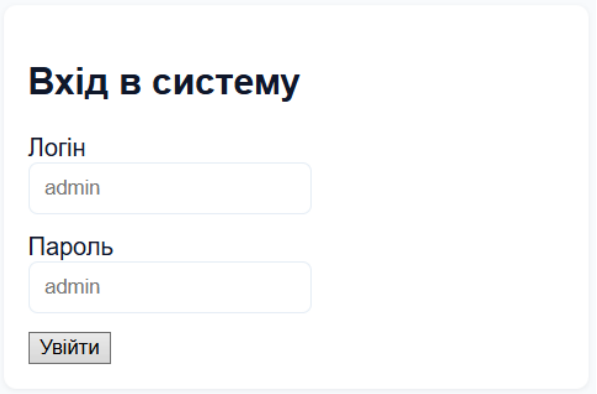

Рисунок 4.1 — Сторінка авторизації

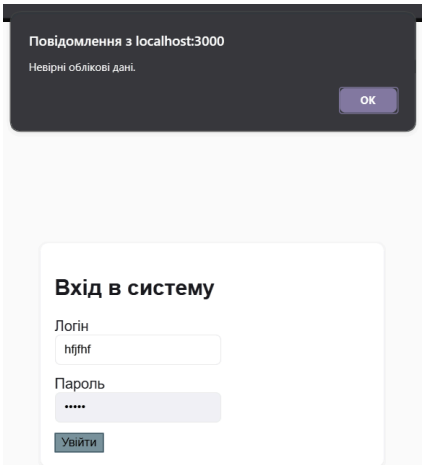

Рисунок 4.2 — Перевірка введення неправильного паролю

Головна сторінка адміністративної панелі є центральним елементом, з якого адміністратор починає роботу. Вона надає загальний огляд стану системи, відображає ключові показники та забезпечує доступ до всіх розділів панелі. На головній сторінці можна побачити короткі статистичні дані, наприклад кількість активних користувачів,

кількість зареєстрованих користувачів за певний період та стан системних процесів (рисунок 4.3).

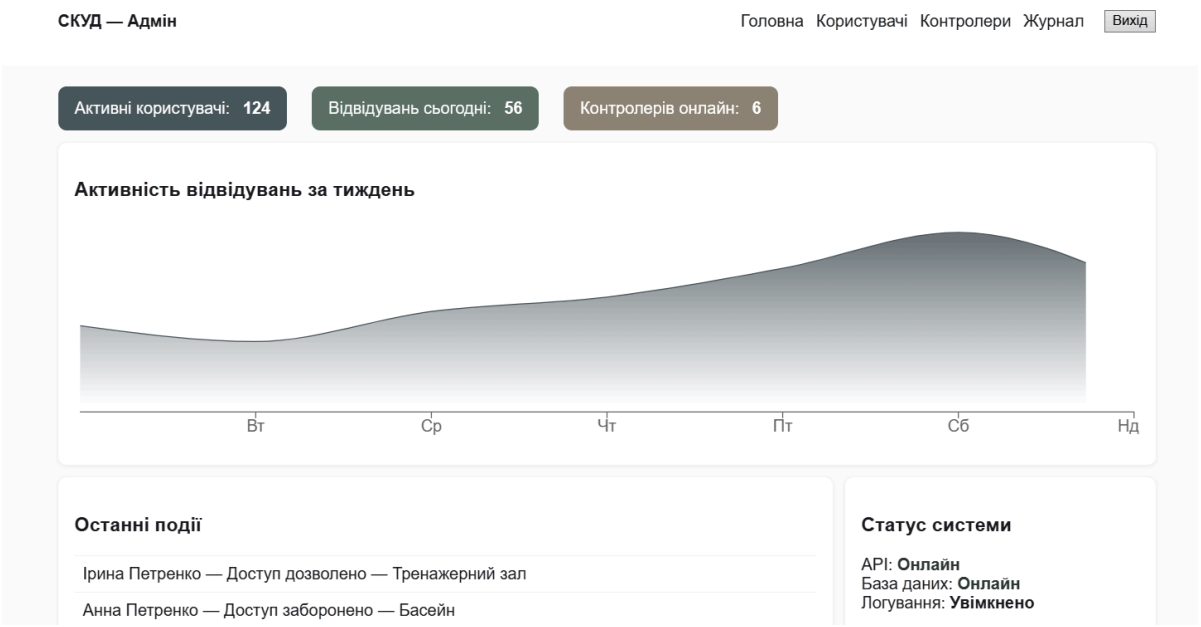

Рисунок 4.3 — Головна сторінка

Головна сторінка адміністративного веб-застосунку виконує роль центру управління та інформування адміністратора. Вона надає зручний огляд стану системи, ключових показників та швидкий доступ до основних функцій. Відразу після авторизації адміністратор потрапляє на головну сторінку, де всі елементи розташовані логічно і інтуїтивно зрозуміло.

У верхній частині сторінки розташована панель навігації, яка містить логотип системи та назву веб-застосунку. Тут також знаходиться меню користувача, де адміністратор може вийти з системи, переглянути свій профіль або перейти до налаштувань. Додатково на панелі можуть бути кнопки сповіщень, що інформують про нові події або важливі зміни у системі. Ця панель завжди доступна і забезпечує швидкий доступ до основних дій без необхідності переходу на інші сторінки.

Зліва від головного робочого простору знаходиться бокове меню. Воно містить посилання на всі основні розділи веб-застосунку: «Головна», «Користувачі», «Статистика», «Налаштування», «Журнал подій». Бокове меню дозволяє адміністратору швидко переходити до потрібних модулів і забезпечує зручну навігацію

по системі. У разі потреби меню можна згорнути, щоб звільнити більше простору для основного контенту.

У центрі головної сторінки розташовані панелі з ключовими показниками. Це невеликі картки або віджети, які показують найважливішу інформацію про роботу системи. Наприклад, можна побачити загальну кількість користувачів, кількість активних користувачів, кількість нових реєстрацій за останній період, стан активних процесів або завдань. Кожна картка оформлена з іконкою або кольоровим індикатором, що допомагає швидко оцінити ситуацію без необхідності детального перегляду всіх даних.

Нижче розташовані графіки та діаграми, які наочно демонструють динаміку роботи системи. Як бачимо, це лінійні графіки активності користувачів за часом, гістограми популярних дій або кругові діаграми розподілу користувачів за ролями. Графіки оновлюються автоматично, що дозволяє адміністратору бачити актуальні дані в режимі реального часу.

Крім того, на головній сторінці є блоки з короткими повідомленнями або сповіщеннями, що інформують про важливі події у системі. Наприклад, адміністратор може одразу дізнатися про нові реєстрації, виконані або відкладені завдання, зміни в налаштуваннях. Це дозволяє швидко реагувати на ситуації, не переходячи у глибокі розділи системи.

Завдяки такому розташуванню та логічному поділу інформації головна сторінка забезпечує адміністратору зручний контроль над усіма аспектами системи**,** швидкий доступ до найважливіших даних та ефективну навігацію між розділами. Всі елементи сторінки протестовані на функціональність: кнопки реагують на натискання, дані відображаються актуально, а навігаційні посилання забезпечують коректний перехід між розділами.

На панелі користувачів адміністратор може переглядати список зареєстрованих користувачів, додавати нових, редагувати дані існуючих користувачів та видаляти непотрібні записи (рисунок 4.3).

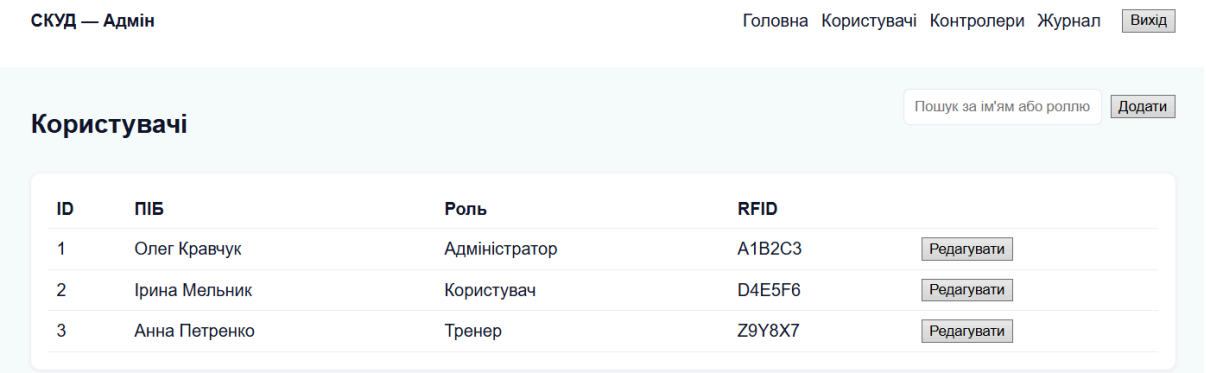

Рисунок 4.3 — Панель користувачів

Під час функціонального тестування перевіряється, що при натисканні кнопки «Додати користувача» відкривається спеціальна форма для введення даних, і після заповнення всіх полів та підтвердження системою дані успішно записуються у базу. Важливо, що після збереження даних адміністратор відразу бачить нового користувача у списку, а база даних підтверджує успішність операції (рисунок 4.4).

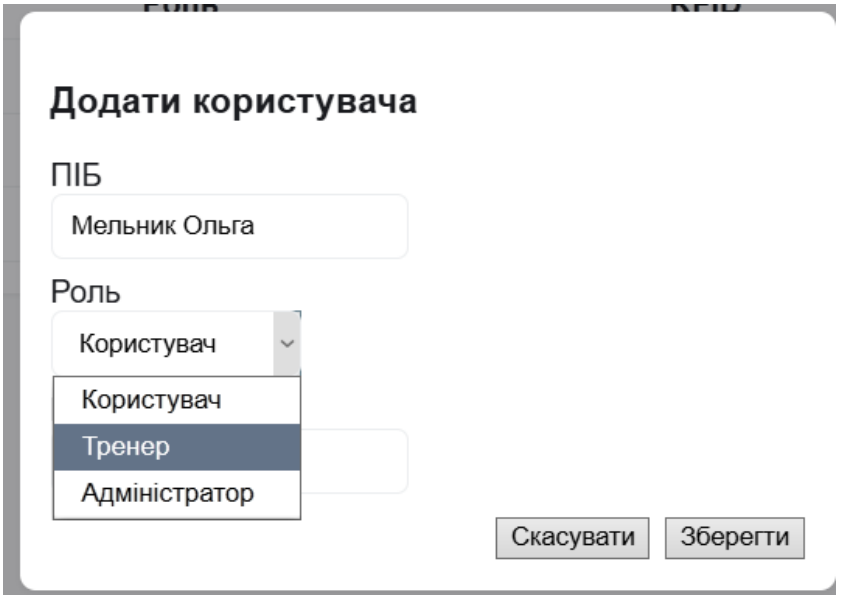

Рисунок 4.4 — Форма для додавання нових користувачів

Якщо адміністратор вирішує змінити дані користувача, натискання кнопки «Редагувати» відкриває вже заповнену форму, де можна змінити будь-який параметр (рисунок 4.5). Після збереження змін система оновлює дані у таблиці та в базі, і адміністратор миттєво бачить результат. У випадку видалення користувача натискання

кнопки «Видалити» викликає підтверджувальне повідомлення; лише після підтвердження запис видаляється з бази і зникає зі списку користувачів.

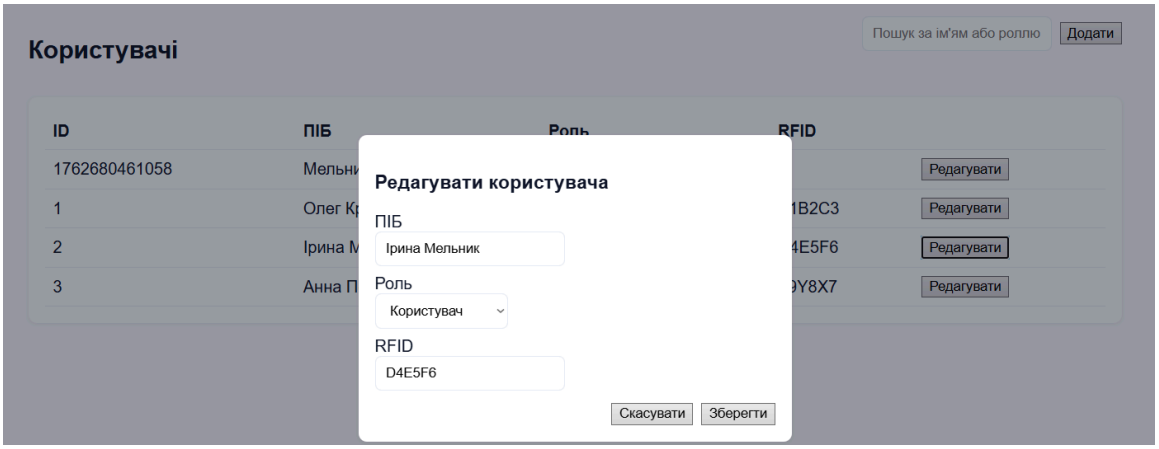

Рисунок 4.5 — Вікно для редагування користувача

Крім того, тестується пошук і фільтрація користувачів — система коректно відображає лише ті записи, що відповідають введеним критеріям (рисунок 4.6).

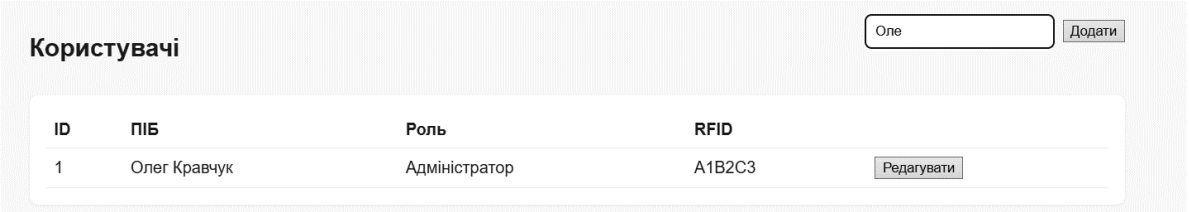

Рисунок 4.6 — Пошук користувача

Панель статистики надає адміністратору можливість переглядати активність користувачів, результати роботи системи та інші ключові показники у вигляді графіків, таблиць та діаграм (рисунок 4.7). Під час тестування перевіряється, що дані, отримані з бази даних, точно відображаються на екрані. Адміністратор може змінювати період відображення даних, і система миттєво оновлює графіки та таблиці. Також перевіряється функція експорту даних у CSV — після натискання кнопки «Експорт» файл формується правильно і містить всю необхідну інформацію. Для перевірки стабільності системи тестуються сценарії з великою кількістю даних, щоб упевнитися, що таблиці і графіки відображаються коректно та не знижують продуктивність вебзастосунку.

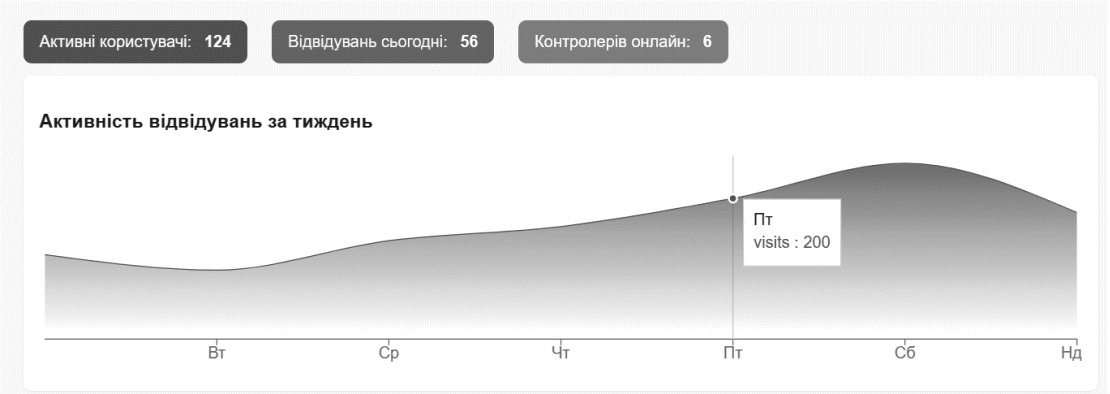

Рисунок 4.7 — Панель статистики адміністратора

Журнал подій є важливим інструментом для контролю та моніторингу роботи веб-застосунку. Він відображає всі ключові дії, які виконуються в системі, і дозволяє адміністратору відстежувати зміни, що відбуваються у роботі користувачів та системи в цілому.

На сторінці журналу подій адміністратор бачить список записів, де кожен рядок відповідає окремій дії (рисунок 4.8). Для кожного запису вказується час виконання дії**,** користувач, який її виконав**, та** короткий опис події**,** наприклад: «додано нового користувача», «змінено роль користувача», «видалено запис» або «експортовано дані». Така структура дозволяє швидко зрозуміти, що відбувається у системі, і за необхідності відновити інформацію про минулі дії.

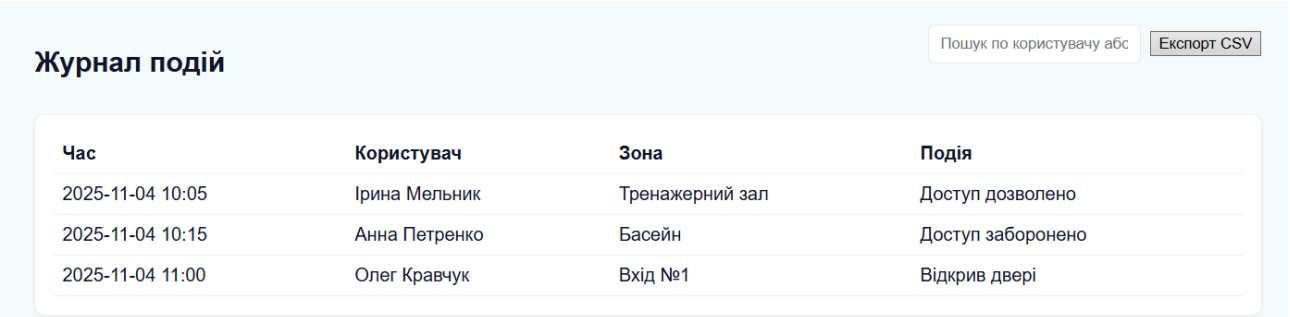

Рисунок 4.8 — Журнал подій

Журнал подій організований так, щоб адміністратор міг легко знаходити потрібну інформацію. Є можливість фільтрувати записи за типом дії**,** сортувати їх за датою або пошукувати конкретні події за іменем користувача. Це дає змогу швидко відслідковувати критичні зміни та виявляти потенційні проблеми в роботі системи.

Під час функціонального тестування журналу подій перевіряється, що всі дії адміністратора та користувачів коректно фіксуються у системі. Наприклад, після додавання нового користувача адміністратор може відкрити журнал і побачити запис про цю дію із зазначенням часу та відповідального користувача. Аналогічно фіксуються редагування, видалення та будь-які зміни в налаштуваннях. Також перевіряється коректність відображення записів при великій кількості подій, щоб впевнитися, що журнал залишається зручним і не перевантажується.

Журнал подій також забезпечує прозорість роботи системи. Адміністратор може контролювати всі зміни, бачити історію дій і переконатися, що користувачі дотримуються правил та ролей, визначених у системі. Крім того, при необхідності він може швидко знайти будь-яку подію та отримати детальну інформацію, що дозволяє ефективно аналізувати діяльність системи та приймати рішення на основі достовірних даних.

Після проведення функціонального тестування веб-застосунку admin-dashboard-fixed можна зробити висновок, що система працює стабільно та відповідає всім вимогам. Було перевірено головну сторінку, панель користувачів, панель статистики, налаштування та журнал подій. Кожен елемент інтерфейсу реагує на дії адміністратора очікуваним чином, всі кнопки, посилання та форми працюють коректно, а дані з бази відображаються точно і своєчасно.

Під час тестування адміністратор зміг успішно додавати, редагувати та видаляти користувачів, змінювати налаштування системи та перевіряти актуальні дані на головній сторінці. Усі зміни, що вносилися, коректно фіксувалися у журналі подій, що забезпечує прозорість та контроль роботи системи. Динамічні графіки та показники відображалися без затримок, і робота з великою кількістю даних не спричиняла збоїв або помилок.

Таким чином, тестування показало, що веб-застосунок стабільний, надійний і готовий до експлуатації. Всі функції працюють коректно, інтерфейс є інтуїтивно зрозумілим і дозволяє адміністратору ефективно керувати системою та контролювати її стан. З урахуванням цього можна зробити висновок, що впровадження веб-застосунку

можливо без додаткових суттєвих змін, і він готовий до використання у реальних умовах.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Оцінювання комерційного потенціалу розробки

Метою проведення комерційного та технологічного аудиту є оцінювання комерційного потенціалу впровадження методів та засобів, розробленої системи контролю доступу до спортивного комплексу з використанням мережевих сервісів.

Для проведення технологічного аудиту було залучено 3— х незалежних експертів Вінницького національного технічного університету к.п.н., доцента Добровольську Наталію Вікторівну., к.т.н., доцента Снігура Анатолія Васильовича., к.т.н., доцента Черняка Олександра Івановича з кафедри обчислювальної техніки. Аудит науково-технічної розробки та її комерційного потенціалу проведено за допомогою таблиці 5.1, застосовуючи п’ятибальну шкалу оцінювання за 12— ма критеріями оцінки.

Таблиця 5.1 — Критерії оцінювання комерційного потенціалу розробки

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Критерій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогі	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів

Продовження таблиці 5.1

4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово—промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві

Продовження таблиці 5.1

11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10— ти років	Термін реалізації ідеї від 3— х до 5— ти років. Термін окупності інвестицій більше 5— ти років	Термін реалізації ідеї менше 3— х років. Термін окупності інвестицій від 3— х до 5— ти років	Термін реалізації ідеї менше 3— х років. Термін окупності інвестицій менше 3— х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

В таблиці 5.2 наведено результати оцінювання науково-технічного рівня і комерційного потенціалу розробки.

Таблиця 5.2 — Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта		
	1. Добровольська Н. В.	2. Снігур А. В.	3. Черняк О. І.
	Бали, виставлені експертами:		
1	2	3	2
2	1	2	4
3	4	3	2
4	3	1	3
5	2	0	3

Продовження таблиці 5.2

6	3	3	4
7	1	4	4
8	2	3	2
9	3	2	3
10	4	3	3
11	3	3	1
12	2	3	3
Сума балів	$СБ_1$=30	$СБ_2$=30	$СБ_3$=34
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_1^3 СБ_i}{3} = \frac{30+30}{3}$		

В таблиці 5.3 наведено шкалу оцінки комерційного потенціалу розробки.

Таблиця 5.3 — Рівні комерційного потенціалу розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0— 10	Низький
11— 20	Нижче середнього
21— 30	Середній
31— 40	Вище середнього
41— 48	Високий

За результатами розрахунків, наведених в таблиці 5.2 та шкалою оцінки наведеної в таблиці 5.3 можна зробити висновок щодо рівня комерційного потенціалу розробки. Середньоарифметична сума балів, виставлених експертами склала 31.3, що відповідає рівню «вище середнього».

Досягнення високого комерційного потенціалу відбулося завдяки значному зниженню витрат ресурсів і часу, витрачених на проведення тестування знань.

В якості аналога для розробки веб-інтерфейсу адміністратора було обрано Kerio Control — популярну систему моніторингу мережевої активності та управління доступом. Цей продукт надає широкі можливості для контролю користувачів, ведення журналів подій та налаштування політик доступу.

Серед недоліків рішення можна відзначити складність інтерфейсу для початківців, надмірну кількість параметрів, що ускладнює навігацію, а також потенційне перевантаження адміністратора інформацією. Крім того, налаштування системи під конкретні сценарії використання може вимагати значного часу.

У розроблюваній системі ці обмеження враховано: інтерфейс спрощено, функції розподілено за логічними модулями, а інформація подається у структурованому вигляді, що забезпечує зручність використання та швидке реагування на події у спортивному комплексі.У таблиці 5.4 наведені основні технічні показники аналога і нового програмного продукту

Таблиця 5.4 — Основні технічні показники аналога і нового програмного продукту

Показники	Аналог	Нова розробка	Відношення параметрів нової розробки до параметрів аналога
Надійність	95%	95%	1/1
Сумісність	96%	96%	1/1
Супровід	90%	91%	1/1
Брендування	85%	85%	1/1
Зв'язок з HRM системою	45%	95%	1/2

Отож, з отриманих результатів у таблиці 5.4 можна зробити висновок що існує зацікавленість серед деяких сторін у нововведеннях розроблених в наслідок виконання роботи.

5.2 Прогнозування витрат на виконання науково-дослідної роботи

Витрати на здійснення науково-дослідної роботи розраховуються за наступними категоріями: витрати на оплату праці, відрахування на соціальні заходи, матеріали, програмне забезпечення для наукових робіт, накладні (загальновиробничі) витрати та ін. Обрахуємо витрати за кожною категорією.

Заробітна плата кожного із залучених осіб визначається за такою формулою:

$$З_0 = \sum_{i=1}^{K} \frac{M_{ni} * t_i}{T_p} \text{ (грн)}, \quad (5.1)$$

де k — кількість посад працівників, залучених до процесу дослідження і розробки;

M_{ni} — місячний посадовий оклад конкретного розробника (інженера, дослідника, науковця тощо), грн.;

$Т_р$ — число робочих днів в місяці; приблизно $Т_р = 22$;

t — кількість робочих днів роботи працівника.

Для проектування і розробки веб-додатку було залучено таких працівників: веб-розробник, дизайнер та тестувальник. Посадові оклади, число днів роботи та витрати на компенсацію наведено в таблиці 5.5.

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт розраховують за формулою:

$$З_р = \sum_{i=1}^{n} C_i \bullet t_i, \quad (5.2)$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника на виконання певної роботи, год.

Таблиця 5.5 — Заробітна плата дослідника в науковій установі

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату грн.
Розробник ПЗ	40 000	1 818,18	17	30 909
Веб-дизайнер	28 000	1 272,72	12	15 272,72
Тестувальник	20 000	909,10	5	4 545,5
Всього				50 727,22

Погодинну тарифну ставку робітника відповідного розряду $С_і$ можна визначити за формулою:

$$C_i = \frac{М_м \cdot К_і \cdot К_с}{Т_р \cdot t_{зм}}, \qquad (5.3)$$

де $М_М$ – розмір прожиткового мінімуму працездатної особи або мінімальної місячної заробітної плати (залежно від діючого законодавства), грн;

$К_і$ – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду;

$К_с$ – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об’єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

$Т_р$ – середня кількість робочих днів в місяці, приблизно Тр = 21…23 дні;

$t_{зм}$ – тривалість зміни, год.

Таблиця 5.6 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Погодинна тарифна ставка, грн	Величина оплати на робітника, грн
1.Підготовчі	3	1	47,6	142,8
2.Монтажні	2	3	64,3	128,6
3.Інтеграційні	2	5	81,0	323,8
4.Налагоджувальні	5	2	52,4	262
5.Випробувальні	3	4	71,4	214,2
Всього				1071,4

Додаткова заробітна плата $З_д$ всіх робітників, які приймали участь в розробці нового технічного рішення розраховується за формулою (5.2) як 10 — 15 % від основної заробітної плати робітників як премія.

На даному підприємстві додаткова заробітна плата нараховується в розмірі 10% від основної заробітної плати.

$$З_д = (З_о + З_р) * \frac{Н_{дод}}{100\%}, \qquad (5.4)$$

$$З_д = 0,11 * (50\,727,22 + 1071,4) = 5\,697,84 \text{ грн}$$

Нарахування на заробітну плату $Н_{ЗП}$ робітників, які брали участь у виконанні роботи, розраховуються за формулою (5.3):

$$З_н = \left(З_о + З_р + З_д\right) * \frac{Н_{ЗП}}{100} \text{ (грн)} \qquad (5.5)$$

де $З_о$ — основна заробітна плата розробників, грн.;

$З_д$ — додаткова заробітна плата всіх розробників та робітників, грн.;

$Н_{ЗП}$ — ставка єдиного внеску на загальнообов'язкове державне соціальне страхування, % .

Основна ставка єдиного внеску на загальнообов'язкове державне соціальне страхування на 2023 рік — 22 %, тоді:

$$З_{н} = (50\,727,22 + 1071,4 + 5697,84) * 0.22 = 12\,649,22 \text{ (грн)}$$

Розрахуємо витрати на матеріали, пристрої, засоби, які використовують при виготовленні одиниці продукції. Розраховуються, згідно їх номенклатури, за формулою:

$$М = \sum_{i=1}^{n} H_j \bullet Ц_j \bullet К_j - \sum_{i=1}^{n} В_j \bullet Ц_{вj}, \quad (5.6)$$

де Нj – норма витрат матеріалу j-го найменування, кг;

n – кількість видів матеріалів;

Цj – вартість матеріалу j-го найменування, грн/кг;

Кj – коефіцієнт транспортних витрат, (Кj = 1,1 … 1,15);

Вj – маса відходів j-го найменування, кг;

Цвj – вартість відходів j-го найменування, грн/кг.

Проведені розрахунки зведені в таблицю 5.7.

Розрахуємо витрати на програмне забезпечення, яке необхідне для проектування та розробки веб-додатку. Балансову вартість програмного забезпечення розраховують за формулою 5.5:

$$В_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \bullet C_{пргі} \bullet К_i, \quad (5.7)$$

де $Ц_{iпрг}$ — ціна придбання/використання одиниці програмного засобу цього виду, грн;

$С_{пргі}$ — кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

$К_і$ — коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо ($К_і$ = 1,10…1,12).

k — кількість найменувань програмних засобів.

Таблиця 5.7 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 од./1 кг, грн	Норма витрат, шт	Вартість витраченого матеріалу, грн
Контролер доступу (Ethernet, RFID, модель типу Wiegand Controller)	950	1	950
Зчитувач RFID-карт (13.56 MHz)	420	2	840
RFID-картки тестові	25	10	250
Мережевий кабель UTP Cat5e (1 м)	12	15	180
Блок живлення 12В для контролера	180	1	180
Wi-Fi модуль ESP32 (для інтеграції та тестів мережевих сервісів)	250	1	250
Raspberry Pi (тестовий шлюз контролерів)	2100	1	2100
Хмарний сервер (тестовий тариф, 1 місяць)	150	1	150
Електроенергія для роботи обладнання (кВт·год)	5	15	75
Канцелярія (папір, маркери, стікери для схем і документування)	200	1	200
Підсумкова вартість матеріалів			5 175
Коефіцієнт транспортування / логістики (10%)			+ 517.5

Загальна вартість з урахуванням транспортування			5 693 грн

Отримані результати наведено в таблиці 5.8.

Таблиця 5.8 — Витрати на використання програмних

Найменування устаткування	Час використання, місяців	Ціна за місяць, грн	Вартість, грн
Підписка Visual Studio Code	2	2200	4400
Підписка на мережеві сервіси	2	1700	3400
Всього			7800

Розрахуємо амортизаційні відрахування по кожному виду обладнання, устаткування яке використовувалось для проектування та розробки системи адаптивного тестування знань.

$$А_{обл} = \frac{Ц_б}{Т_в} * \frac{t_{вик}}{12}, \quad (5.8)$$

де $Ц_б$ — балансова вартість даного виду обладнання (приміщень), грн.;

$t_{вик}$ — час користування;

$Т_в$ — термін використання обладнання (приміщень), цілі місяці.

Амортизаційні відрахування наведено в таблиці 5.9.

Таблиця 5.9 — Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Ноутбук	35000	2	2	2916,66
Wi-Fi/Bluetooth	4000	2	2	333,33

маршрутизатор				
Всього				3250

Витрати на енергію визначаються на основі витрат на одиницю продукції та тарифів на енергію за допомогою формули:

$$В_е = В \cdot П \cdot Ф \cdot К_п \text{ [грн]}, \qquad (5.9)$$

де В — вартість 1кВт електроенергії;

П — установлена потужність обладнання, кВт;

Ф — фактична кількість годин роботи комп'ютера при створенні програмного продукту, годин;

$К_п$ — коефіцієнт використання потужності.

Отже, витрати на енергію становлять:

$$В_е = 4,32 \cdot 0,5 \cdot 270 \cdot 0,5 = 291,6 (грн).$$

Витрати за доступ до Інтернет можна розрухувати за формулою:

$$В_{ді} = Ц_{ді} \cdot Т \text{ [грн]}, \qquad (5.10)$$

де $Ц_{ді}$ — це ціна доступу за місяць;

Т — кількість місяців використання доступу до мережі.

$$В_{ді} = 230 \cdot 3 = 690,00 (грн).$$

Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати робітників за формулою 5.11.

$$B_{ін} = З_о \cdot 100\% \text{ [грн]}. \qquad (5.11)$$

$$B_{ін} = 50\ 727,22 \cdot 0,5 = 25\ 363,61 \text{ (грн)}.$$

Дана стаття витрат охоплює витрати на управління організацією, оплату службових відряджень, витрати на утримання, ремонт та експлуатацію основних засобів, витрати на опалення, освітлення, водопостачання, охорону праці тощо. Витрати за статтею «Службові відрядження» розраховуються як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$\text{В}_{\text{св}} = \left(\text{З}_{\text{о}} + \text{З}_{\text{р}}\right) * \frac{\text{Н}_{\text{св}}}{100\%}, \qquad (5.12)$$

де $\text{Н}_{\text{св}}$ – норма нарахування за статтею «Службові відрядження».

$$\text{В}_{\text{св}} = (50727,22 + 1071,4) * 0.2 = 10\ 359,724 \text{ грн}$$

Накладні (загальновиробничі) витрати Внзв охоплюють: витрати на управління організацією, оплата службових відряджень, витрати на утримання, ремонт та експлуатацію основних засобів, витрати на опалення, освітлення, водопостачання, охорону праці тощо. Накладні (загальновиробничі) витрати Внзв можна прийняти як (100…150)% від суми основної заробітної плати розробників та робітників, які виконували дану МКНР, тобто:

$$\text{В}_{\text{нзв}} = \left(\text{З}_{\text{о}} + \text{З}_{\text{р}}\right) \bullet \frac{\text{Н}_{\text{нзв}}}{100\%}, \qquad (5.13)$$

де $\text{Н}_{\text{нзв}}$ – норма нарахування за статтею «Інші витрати».

$$\text{В}_{\text{нзв}} = (50727,22 + 1071,4) \bullet \frac{100}{100\%} = 51\ 798,62 \text{грн}$$

Сума всіх статей витрат дає в результаті витрати на проведення дослідження та розробку адаптивної системи тестування знань і розраховується за формулою:

$$В = З_о + З_р + З_{дод} + З_н + М + В_{прг} + А_{обл} + В_е + В_{ін} + В_{ДІ} + В_{св} + В_{нзв}$$

$$В = 50727,22 + 1071,4 + 5697,84 + 12649,22 + 5693 + 7800 + 3250 + 291,$$
$$= 175\ 392,23$$

Загальні витрати ЗВ на завершення роботи та оформлення її результатів розраховуються за формулою:

$$ЗВ = \frac{В}{\eta}, \qquad (5.14)$$

де η — коефіцієнт, який характеризує стадію виконання даної НДР.

Так, якщо розробка знаходиться:

— на стадії науково— дослідних робіт, то $\beta \approx 0,1$;

— на стадії технічного проектування, то $\beta \approx 0,2$;

— на стадії розробки конструкторської документації, то $\beta \approx 0,3$;

— на стадії розробки технологій, то $\beta \approx 0,4$;

— на стадії розробки дослідного зразка, то $\beta \approx 0,5$;

— на стадії розробки промислового зразка, $\beta \approx 0,7$;

— на стадії впровадження, то $\beta \approx 0,9$.

Звідси:

$$ЗВ = \frac{175\ 392,23}{0.7} = 250\ 560,32 \text{ грн.}$$

Витрати на виконання наукової роботи та впровадження її результатів становитиме 250 560,32 грн.

5.3 Розрахунок економічної ефективності науково-технічної розробки

Економічна ефективність дозволяє спрогнозувати чистий прибуток, який може бути отриманий від впровадження розробленої системи.

При розрахунку економічної ефективності потрібно обов'язково враховувати зміну вартості грошей у часі, оскільки від вкладення інвестицій до отримання прибутку минає чимало часу.

Для розрахунку збільшення чистого прибутку підприємства $\Delta П_i$, для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки використовується формула формулою 5.11:

$$\Delta П_i = \sum_{1}^{n} \left(\Delta Ц_o \cdot N + Ц_o \cdot \Delta N \right)_i \cdot \lambda \cdot \rho \cdot \left(1 - \frac{\upsilon}{100} \right) \qquad (5.15)$$

де $\Delta Ц_0$ – покращення основного оціночного показника від впровадження результатів розробки у даному році.

N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки:

$Ц_o$ – основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки:

л – коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт *л* = 0,8333.

р – коефіцієнт, який враховує рентабельність продукту. р = 0,25;

х – ставка податку на прибуток. У 2025 році – 18%.

Припустимо, що ціна зросте на 1000 грн. Кількість одиниць реалізованої продукції також збільшиться: протягом першого року на 30 шт., протягом другого року – на 40 шт., протягом третього року на 60 шт. Реалізація продукції до впровадження

розробки складала 1 шт., а її ціна до 45000 грн. Розрахуємо прибуток, яке отримає підприємство протягом трьох років.

$$\Delta П_1 = [1000 \cdot 1 + (45000 + 1000) \cdot 30] \cdot 0,833 \cdot 0,25 \cdot \left(1 - \frac{18}{100}\right) = 235\,563 \text{ грн.}$$

$$\Delta П_2 = [1000 \cdot 1 + (45000 + 1000) \cdot (30 + 40)] \cdot 0,833 \cdot 0,25 \cdot \left(1 - \frac{18}{100}\right) = 549\,619 \text{ грн.}$$

$$\Delta П_3 = [1000 \cdot 1 + (45000 + 1000) \cdot (30 + 40 + 60)] \cdot 0,833 \cdot 0,25 \cdot \left(1 - \frac{18}{100}\right) = 1\,021\,5$$

Далі за формулою 5.12 розраховують приведену вартість збільшення всіх чистих прибутків ПП, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (5.16)$$

де ΔПі — збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково— технічної розробки, грн;

Т — період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,2;

t – період часу (в роках).

$$ПП = \frac{235\,563}{(1+0,2)^1} + \frac{549\,619}{(1+0,2)^2} + \frac{1\,021\,537}{(1+0,2)^3} = 1\,169\,000 \text{ грн.}$$

Отже, розрахунки показують, що комерційний ефект від впровадження розробки виражається у значному збільшенні чистого прибутку підприємства.

Основними показниками, які визначають доцільність фінансування наукової розробки інвестором, є абсолютна і відносна ефективність вкладених інвестицій та термін їх окупності.

Якщо $E_{абс} > 0$, то результат від проведення наукових досліджень та їх впровадження принесе прибуток, але це також ще не свідчить про те, що інвестор буде зацікавлений у фінансуванні даного проекту (роботи).

Розрахуємо абсолютну ефективність інвестицій, вкладених у реалізацію проекту. Ставка дисконтування τ дорівнює 0,1. Отримаємо:

Розраховуємо величину початкових інвестицій PV, які розробник (замовник) має вкласти для здійснення науково— технічної розробки.

Для цього можна використати формулу:

$$PV = k_{розр} \cdot ЗВ \text{ [грн]}, \qquad (5.17)$$

де розр *k* — коефіцієнт, що враховує витрати розробника (замовника) на впровадження науково— технічної розробки. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай розр k =2...5, але може бути і більшим;

ЗВ — загальні витрати на проведення науково— технічної розробки та оформлення її результатів, грн.

$$PV = 2 \cdot 250\,560,32 = 501\,120,64 \text{ грн}$$

Тоді абсолютний економічний ефект $E_{абс}$ або чистий приведений дохід (NPV, Net Present Value) для потенційного інвестора від можливого впровадження та комерціалізації розробки становитиме:

$$E_{абс} = ПП - PV, \qquad (5.18)$$

де ПП — приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково— технічної розробки, грн;

PV — теперішня вартість початкових інвестицій, грн.

$$E_{\text{абс}} = 1\,169\,000 - 501\,120,64 = 667\,879,36 \text{ грн.}$$

Оскільки $E_{абс} > 0$ то вкладання коштів на виконання та впровадження результатів НДДКР може бути доцільним.

5.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності

Для остаточного прийняття рішення про впровадження розробки та виведення її на ринок необхідно розрахувати внутрішню економічну дохідність $E_в$ або показник внутрішньої норми дохідності (IRR, Internal Rate of Return) вкладених інвестицій та порівняти її з так званою бар'єрною ставкою дисконтування, яка визначає ту мінімальну внутрішню економічну дохідність, нижче якої інвестиції в будь— яку розробку вкладати буде економічно недоцільно.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$. Для цього користуються формулою 5.15:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \qquad (5.19)$$

де $E_{абс}$ — абсолютний економічний ефект вкладених інвестицій, грн; PV — теперішня вартість початкових інвестицій, грн;

$T_ж$ — життєвий цикл науково— технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, роки.

$$\text{E}_\text{в} = \sqrt[3]{1 + \frac{667\,879,36}{501\,120,64}} - 1 = 0.38 = 38\%$$

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається за формулою 5.16:

$$\tau = d + f, \qquad (5.20)$$

де d — середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні d= (0,14…0,2);

f – показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,1).

$$\tau_{min} = 0.18 + 0,05 = 0.23$$

Так як $E_в > \tau_{min}$ то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Далі розраховується період окупності інвестицій, вкладених у реалізацію проекту за формулою 5.17.

$$T_{ок} = \frac{1}{E_в}, \qquad (5.21)$$

де Ев — внутрішня економічна дохідність вкладених інвестицій.

$$\text{Т}_{\text{ок}} = \frac{1}{0.,38} = 2,6 \text{ роки}$$

Так як $T_{ок} \leq 3...5$-ти років, то фінансування даної наукової розробки в принципі є доцільним.

5.5 Результати економічного аналізу

В даному розділі було проведено оцінку комерційного потенціалу розробки програмного забезпечення для адаптивного тестування знань. Економічний потенціал склав значення, яке є вище середнього. Також було зпрогнозовано витрати на

проектування та реалізацію системи за необхідними статтями витрат, які склали 250 560,32 грн.

Обрахунок економічної ефективності та терміну окупності вкладених інвестицій показали, що розробка є економічно доцільною та привабливою для потенційних інвесторів. Термін окупності інвестицій складає 2,6 роки.

ВИСНОВКИ

У результаті виконання магістерської роботи було розроблено комп’ютерну систему контролю доступу для спортивних комплексів, яка забезпечує ефективне управління правами користувачів та моніторинг їхньої активності у межах об’єкта. Створена система виступає як сучасне рішення для організацій, що потребують контролю відвідувачів, персоналу та обладнання, а також можливості аналітичної оцінки використання ресурсів. Головною перевагою є інтеграція апаратних засобів контролю доступу, мережевих сервісів та веб-інтерфейсу для адміністратора, що дозволяє в реальному часі відстежувати стан системи та приймати обґрунтовані рішення.

У першому розділі було здійснено аналіз сучасних систем контролю доступу, що застосовуються у спортивних комплексах, та технологій ідентифікації користувачів. Розглянуто RFID-системи, біометричні методи, бездротові протоколи та моделі інтеграції з інфраструктурою закладів. Проведене дослідження дозволило визначити їхні переваги та обмеження, а також сформувати вимоги до майбутньої системи з урахуванням потреб спортивних організацій.

У другому розділі важливим напрямом стало дослідження використання мережевих сервісів як основи для взаємодії між компонентами системи. Було встановлено, що використання REST API, веб-сервісів і клієнт–серверної моделі забезпечує уніфікований протокол обміну даними, можливість масштабування, оперативність обробки подій та інтеграцію з апаратними контролерами. Це дозволило обґрунтувати вибір саме сервісно-орієнтованого підходу у побудові системи.

У третьому розділі сформовано архітектуру комп’ютерної системи контролю доступу, яка складається з апаратного рівня, рівня мережевих сервісів та веб-інтерфейсу адміністратора. Спроєктовано структуру бази даних, моделі користувачів, журналів подій та ролей доступу. Обрана багаторівнева архітектура забезпечує безпеку, надійність, підтримку реального часу та можливість масштабування системи для різних спортивних комплексів.

У межах четвертого розділу було розроблено програмне забезпечення системи з використанням сучасних інструментів веб-розробки. Реалізовано серверну частину з

модулями авторизації, управління ролями, обробки подій контролерів і REST API для взаємодії з обладнанням. Створено веб-інтерфейс адміністратора, що забезпечує зручне керування користувачами, моніторинг подій, перегляд статистики та роботу з даними в режимі реального часу. Проведене тестування підтвердило коректність роботи основних модулів, стабільність функціонування та відповідність системи заявленим вимогам.

У п’ятому розділі було виконано економічні розрахунки, в результаті яких зроблено висновок, що потенційна зацікавленість інвесторів у фінансуванні розробки є досить високою.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1 Системи контролю доступу: що це таке і як працює [Електронний ресурс]. — Режим доступу: https://zakarpattya.net.ua/News/200909-Systemy-kontroliu-dostupu:-shcho-tse-take-i-iak-pratsiuie

2 Системи контролю доступу: різновиди та поради щодо вибору [Електронний ресурс]. — Режим доступу: https://insider-media.net/life/systemy-kontroliu-dostupu-riznovydy-ta-porady-shchodo-vyboru

3 What is Access Control [Електронний ресурс]. — Режим доступу: https://continuumgrc.com/uk/what-is-access-control

4 Система контролю доступу (СКУД): принцип дії, склад, особливості застосування [Електронний ресурс]. — Режим доступу: https://imperia.org.ua/article/sistema-kontrolyu-dostupu-skud-princip-dii-sklad-osoblivosti-zastosuvannya

5 PERCo-Web: система контролю доступу [Електронний ресурс]. — Режим доступу: https://www.perco.com/products/perco-web-access-control-system

6 Інформаційна безпека: системи контролю доступу [Електронний ресурс]. — Режим доступу: http://www.infobezpeka.com/publications/?id=92

7 Системи контролю доступу [Електронний ресурс]. — Режим доступу: https://studfile.net/preview/5129599

8 Перший крок впровадження системи контролю та управління доступом [Електронний ресурс]. — Режим доступу: https://www.krruda.dp.ua/pershij-krok-vprovadzhennya-sistemi-kontrolyu-ta-upravlinnya-dostupom

9 Як працює система контролю доступу (СКД або СКУД) [Електронний ресурс]. — Режим доступу: https://u-prox.systems/yak-praczyuye-sistema-kontrolyu-dostupu-skd-abo-skud

10 Node.js [Електронний ресурс]. — Режим доступу: https://en.wikipedia.org/wiki/Node.js

11 MySQL [Електронний ресурс]. — Режим доступу: https://www.mysql.com

12 REST API: що це, як працює, переваги і приклади [Електронний ресурс]. — Режим доступу: https://goit.global/ua/articles/rest-api-shcho-tse-iak-pratsiuie-perevahy-i-pryklady

13 User Authentication — Clerk [Електронний ресурс]. — Режим доступу: https://clerk.com/user-authentication

14 System Architecture [Електронний ресурс]. — Режим доступу: https://cyphrax.com/system-architecture

15 Контролер ZKTeco InBio160 [Електронний ресурс]. — Режим доступу: https://www.bezpeka-shop.com/ua/product/kontroller-zkteco-inbio160

16 RFID-зчитувач ZKTeco KR500E [Електронний ресурс]. — Режим доступу: https://zktecoua.com/ua/products/rfid-reader-zkteco-kr500e

17 Замок електромагнітний YLI YM-280N [Електронний ресурс]. — Режим доступу: https://lock.ua/catalog/zamki-elektromagnitni/zamok-elektromagnitniy-yli-ym-280n-12-24v-nakladniy-280-kg-.html

18 Електроживлення 12В, 3А [Електронний ресурс]. — Режим доступу: https://hard.rozetka.com.ua/ua/36064/p36064

19 Тестування програмного забезпечення [Електронний ресурс]. — Режим доступу: https://studfile.net/preview/9612104/page:7

20 Client–Server Architecture [Електронний ресурс]. — Режим доступу: https://training.qatestlab.com/blog/technical-articles/client-server-architecture

21 Схеми баз даних: основи проєктування [Електронний ресурс]. — Режим доступу: https://foxminded.ua/skhemy-bazy-danyh

22 Що таке REST API [Електронний ресурс]. — Режим доступу: https://foxminded.ua/shcho-take-rest-api

23 Найкращі практики проєктування REST API [Електронний ресурс]. — Режим доступу: https://devzone.org.ua/post/naykrashchi-praktyky-proyektuvannia-rest-api

24 REST API: основи та приклад реалізації на Python Flask [Електронний ресурс]. — Режим доступу: https://cloud.itstep.org/blog/rest-api-what-is-it-restful-project-on-python-flask

25 User Management — Архітектура системи [Електронний ресурс]. — Режим доступу: https://ddm-architecture-cluster-mgmt.apps.krrt-stage.ncr.gov.ua/ua/platform/1.9.7/arch/architecture/platform/operational/user-management/overview.html

26 Access Control Models and Architectures for IoT and Cyber-Physical Systems / Gupta M., Bhatt S., Alshehri A.H., Sandhu R. — Springer, 2022. — XIV, 173 с.

27 Security and Resilience of Control Systems: Theory and Applications / — Springer, 2022. — LNCIS series.

28 Безпека інформаційних систем : навчальний посібник / Пашорін В. І., Костюк Ю. В. — Державний торговельно-економічний університет, 2023. ur.knute.edu.ua

29 Автоматизована система контролю доступу до виробничих приміщень / Вітер К. М., Лебеденко Ю. О., Гром’як О. А. — матеріали Всеукраїнської конференції, 2023.

30 Методичні вказівки до виконання магістерських кваліфікаційних робіт студентами спеціальності 123 «Комп’ютерна інженерія» [Електронний ресурс] / Уклад. О. Д. Азаров, О. В. Дудник, С. І. Швець. − Вінниця : ВНТУ 2023. − 58 с.

31 Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.

ДОДАТОК А

Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп’ютерної інженерії

Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТ

проф., д.т.н.. Азаров О.Д..

“ 3” жовтня 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи

“ Комп’ютерна система контролю доступу до спортивного комплексу з використанням мережевих сервісів ”

Науковий керівник: доцент

к.т.н. доц. каф.ОТ

_____ Кожем’яко А. В.

Студент групи 2КІ-24м

_____ Доманський А. Ф.

ВНТУ 2025

1 Підстава для виконання магістерської кваліфікаційної роботи (МКР)

1.1 Актуальність теми дослідження полягає в тому, що розроблювана комп'ютерна система контролю доступу до спортивного комплексу з використанням мережевих сервісів забезпечує ефективне управління входом і виходом відвідувачів, а також моніторинг активності користувачів у реальному часі. Така система дозволяє підвищити рівень безпеки, автоматизувати процес ідентифікації та зменшити вплив людського фактора під час перевірки доступу.

1.2 Наказ про затвердження теми МКР.

2 Мета МКР і призначення розробки

2.1 Мета роботи – розширення функціональних можливостей комп'ютерної системи контролю доступу до спортивного комплексу з використанням мережевих сервісів, за рахунок інтеграції локальних пристроїв контролю з центральним сервером через веб-сервіси, що забезпечить автоматизований облік користувачів, підвищений рівень безпеки та ефективне управління доступом до об'єктів інфраструктури.

2.2 Призначення розробки — визначається необхідністю створення системи, яка забезпечить ефективний контроль доступу до приміщень спортивного комплексу та моніторинг активності користувачів. Система дозволяє адміністратору керувати правами доступу відвідувачів і персоналу, відстежувати їх дії в межах об'єкта, а також отримувати статистичні дані у реальному часі за допомогою мережевих сервісів.

3 Вихідні дані для виконання МКР

3.1 Проведення аналізу існуючих методів та принципів.

3.2 Розробка алгоритму роботи веб- додатку.

3.4 Проведення верифікації та аналізу отриманих результатів.

3.5 Виконання розрахунків для доведення доцільності нової розробки з економічної точки зору.

4 Вимоги до виконання МКР

Головна вимога — щоб система забезпечувала інформаційну підтримку адміністратора та користувачів щодо поточного стану доступу й активності відвідувачів спортивного комплексу. Вона повинна відображати в реальному часі інформацію про перебування користувачів у різних зонах об’єкта, надавати статистичні дані щодо відвідуваності, часу перебування та історії входів і виходів.

5 Етапи МКР та очікувані результати

Етапи роботи та очікувані результати приведено в Таблиці А.1.

Таблиця А.1 — Етапи МКР

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Аналіз існуючих рішень			Розділ 1
2	Визначення архітектури системи			Розділ 2
3	Розробка алгоритму та функціоналу комп’ютерної системи			Розділ 3
4	Тестування системи			Розділ 4
4	Підготовка економічної частини			Розділ 5
6	Оформлення пояснювальної записки, графічного матеріалу і презентації			ПЗ, графічний матеріал і презентація
7	Підготовка і підпис супроводжуючих документів, нормоконтроль та тест на плагіат			Оформленні документи

6 Матеріали, що подаються до захисту МКР

До захисту подаються: пояснювальна записка МКР, графічні і ілюстративні матеріали, протокол попереднього захисту МКР на кафедрі, відгук наукового

керівника, відгук опонента, протоколи складання державних екзаменів, анотації до МКР українською та іноземною мовами.

7 Порядок контролю виконання та захисту МКР

Виконання етапів графічної та розрахункової документації МКР контролюється науковим керівником згідно зі встановленими термінами. Захист МКР відбувається на засіданні Екзаменаційної комісії, затвердженої наказом ректора.

8 Вимоги до оформлювання та порядок виконання МКР

8.1 При оформлюванні МКР використовуються:

— ДСТУ 3008: 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302: 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— ГОСТ 2.104–2006 «Єдина система конструкторської документації. Основні написи»;

— методичні вказівки до виконання магістерських кваліфікаційних робіт зі спеціальності 123 — «Комп’ютерна інженерія»;

— документи на які посилаються у вище вказаних.

8.2 Порядок виконання МКР викладено в «Положення про кваліфікаційні роботи на другому (магістерському) рівні вищої освіти СУЯ ВНТУ–03.02.02 П.001.01:21

ДОДАТОК Б

Протокол перевірки кваліфікаційної роботи

Назва роботи: ____ Комп’ютерна система контролю доступу до спортивного комплексу з використанням мережевих сервісів._____

Тип роботи: _____ магістерська кваліфікаційна робота _____

Підрозділ: _____кафедра обчислювальної техніки _____

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism (КП1) 1 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

✔ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак академічного плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту.

У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

У роботі виявлено ознаки академічного плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень. Робота до захисту не приймається.

Експертна комісія:

_____ _____
(прізвище, ініціали, посада) (підпис)

_____ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ ___Захарченко С. М.__
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ _____Кожем’яко А. В._____

Здобувач _____ _____Доманський А. Ф_____

ДОДАТОК В

Багаторівнева архітектура управління доступом та даними в хмарному середовищі

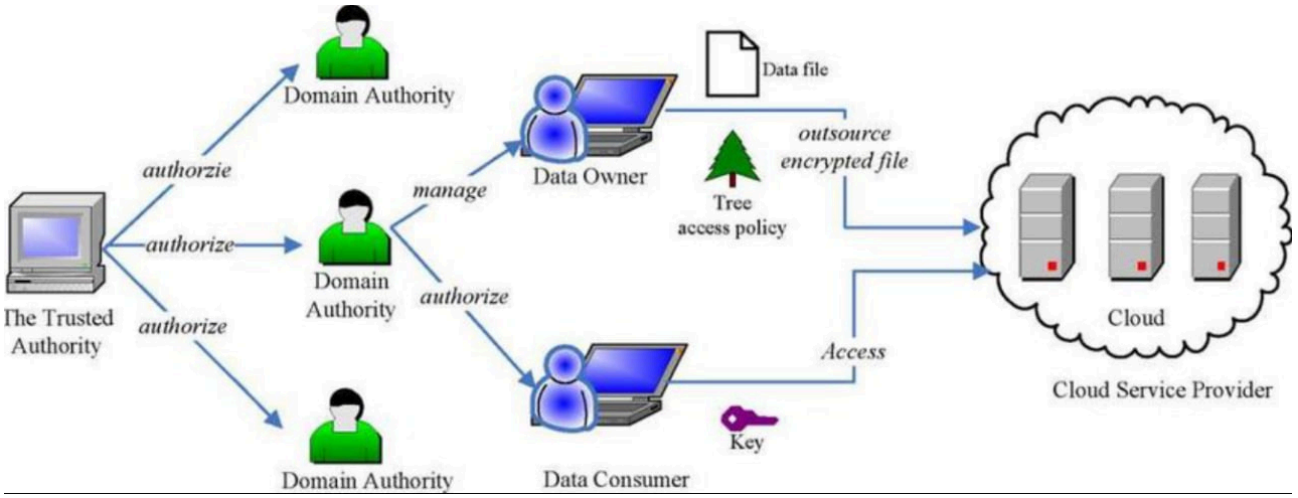

Рисунок В.1 — Схема архітектури управління доступом та даними в хмарному середовищі

ДОДАТОК Г

Блок-схема апаратної частини системи контролю доступу на базі RFID

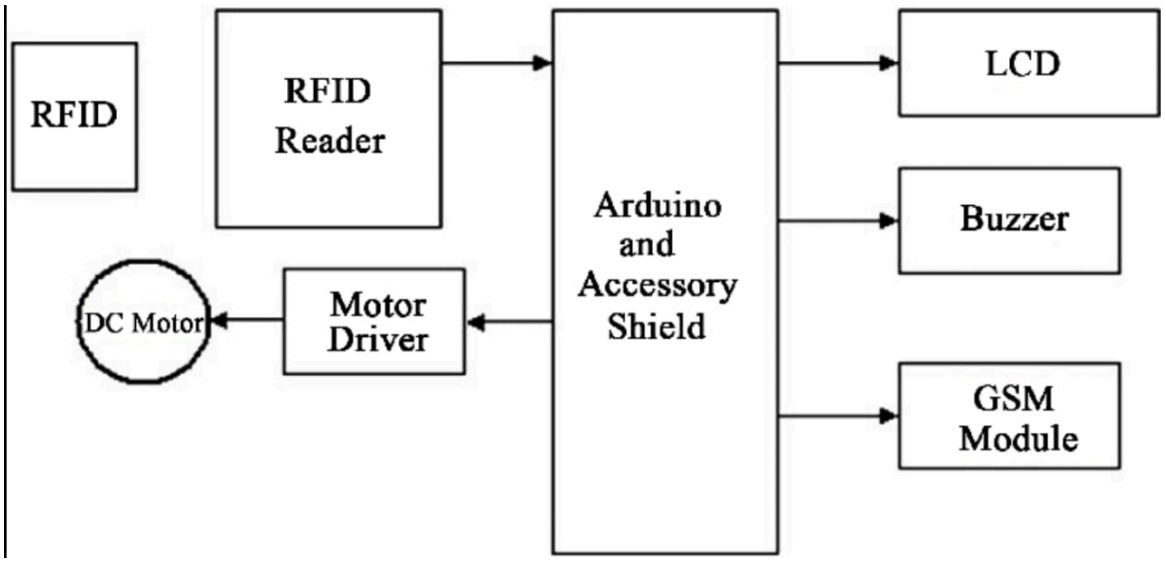

Рисунок Г.1 — Блок-схема апаратної частини системи

ДОДАТОК Д

UML-модель процесу позиціонування

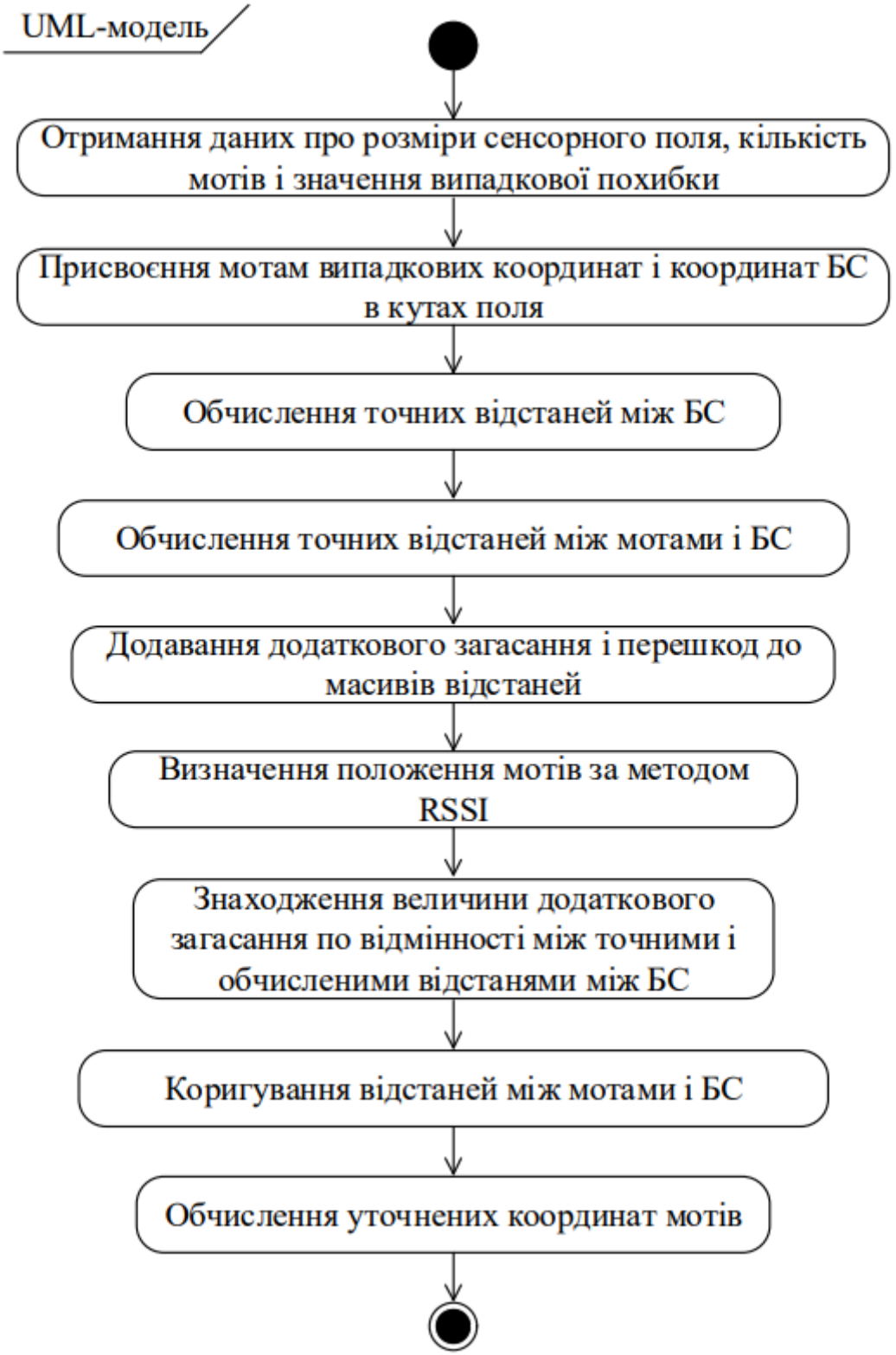

Рисунок Д.1 — UML-модель процесу позиціонування

ДОДАТОК Е

ER-діаграма структури бази даних

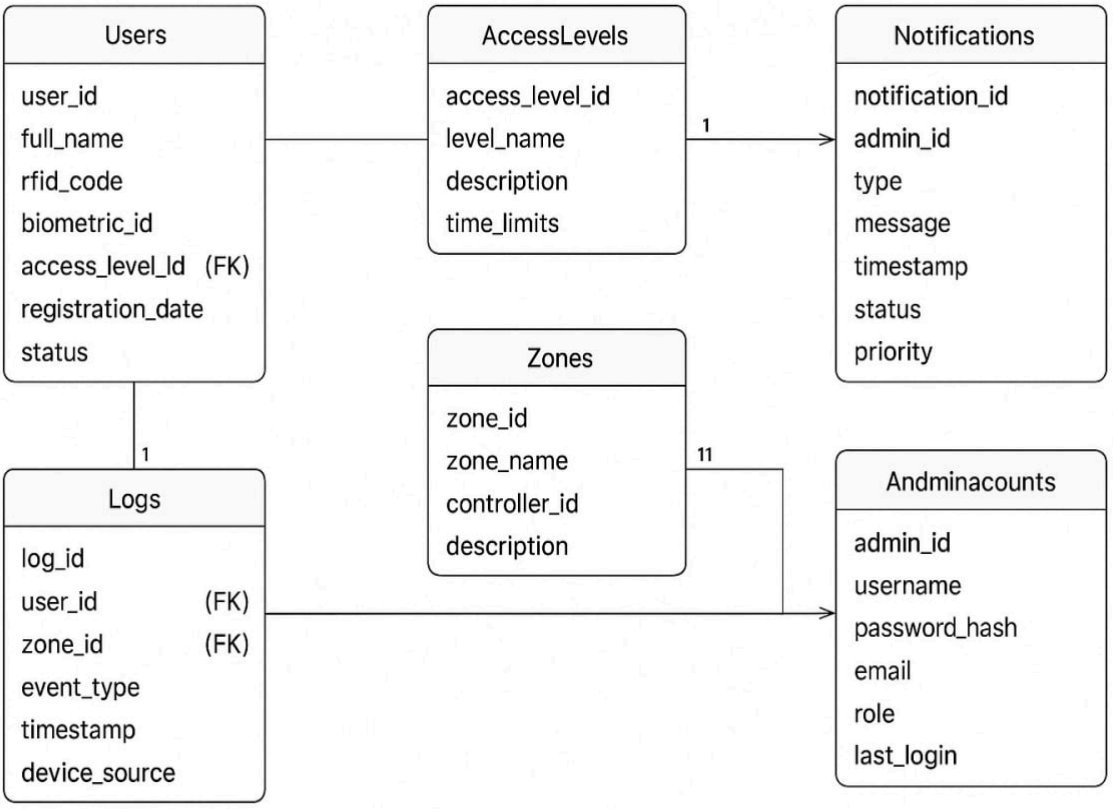

Рисунок Е.1 — ER-діаграма

ДОДАТОК Ж

Структурна схема розподіленої комп'ютерної системи контролю доступу на базі мережі TCP/IP

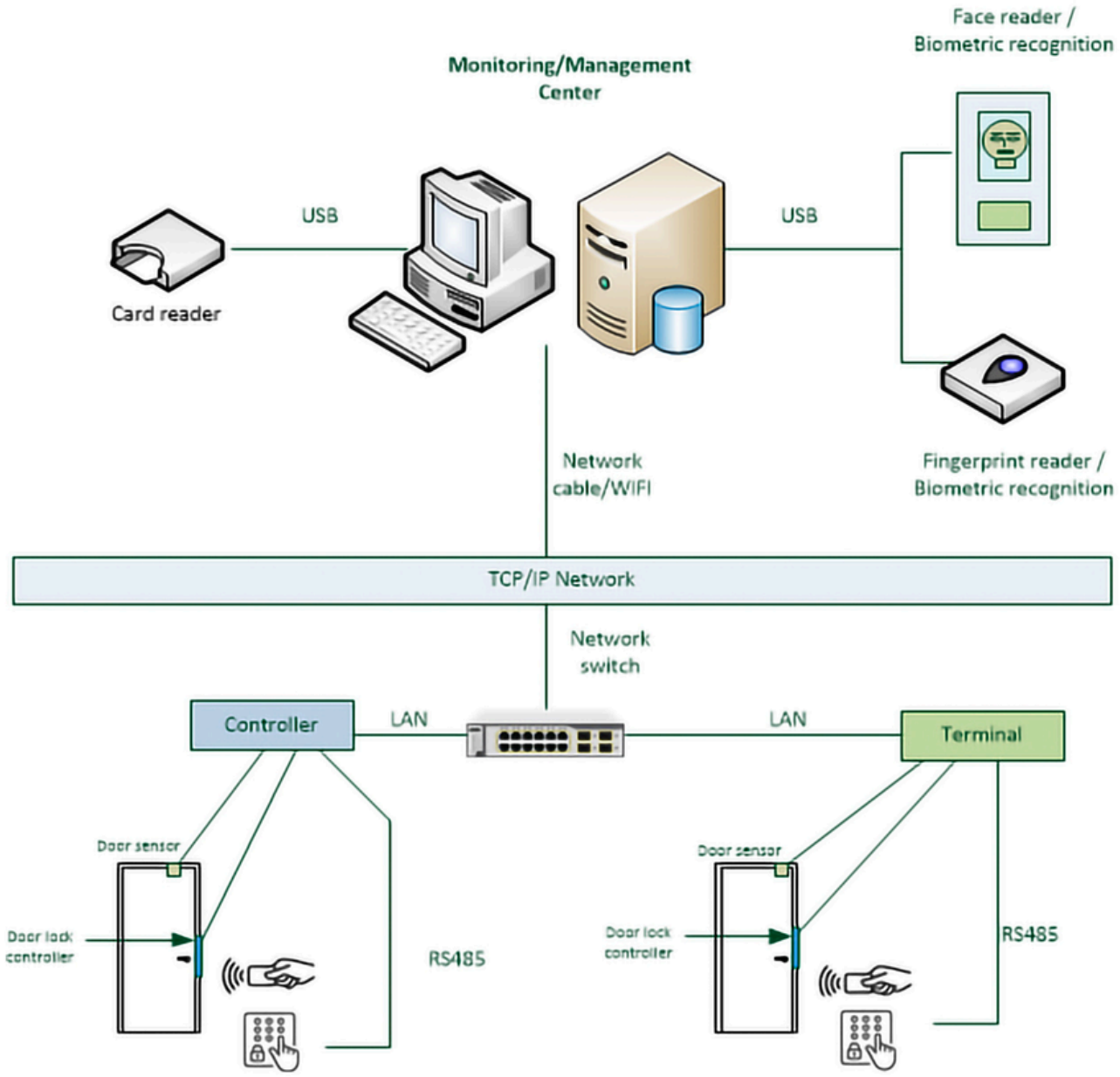

Рисунок Ж.1 — Структурна схема розподіленої комп'ютерної системи контролю доступу на базі мережі TCP/IP

ДОДАТОК И

Блок-схема алгоритму клієнтської частини

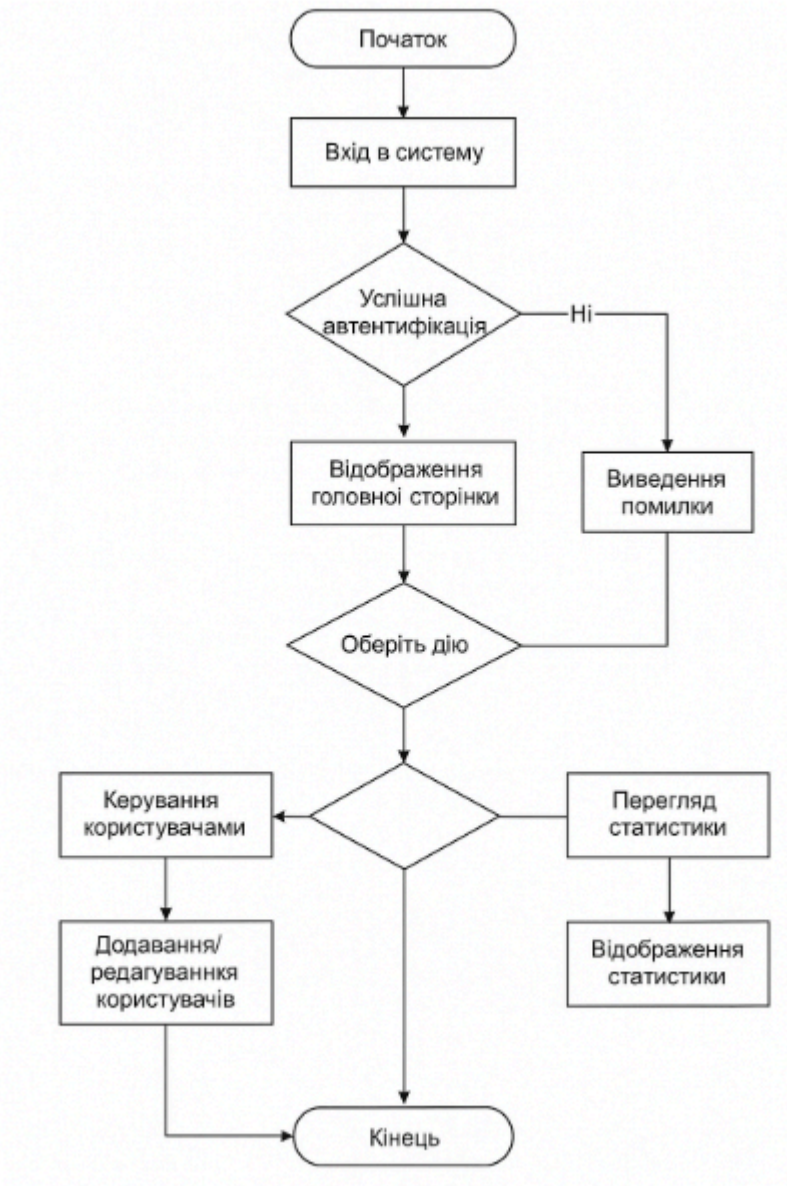

Рисунок И.1 — Блок-схема алгоритму клієнтської частини

ДОДАТОК К

Лістинг програмного забезпечення

Лістинг К.1 — Лістинг файлу DashboardPage.jsx

```
import React from 'react';
import { ResponsiveContainer, AreaChart, Area, XAxis, Tooltip } from 'recharts';
const data = [
  {day:'Пн', visits: 120},
  {day:'Вт', visits: 98},
  {day:'Ср', visits: 140},
  {day:'Чт', visits: 160},
  {day:'Пт', visits: 200},
  {day:'Сб', visits: 250},
  {day:'Нд', visits: 180},
];
export default function DashboardPage(){
  return (
    <div className="container" style={{paddingTop:20}}>
      <div style={{display:'flex', gap:12, marginBottom:12}}>
        <div className="kpi">Активні користувачі: <strong
style={{marginLeft:8}}>124</strong></div>
        <div className="kpi" style={{background:'#10b981'}}>Відвідувань сьогодні:
<strong style={{marginLeft:8}}>56</strong></div>
        <div className="kpi" style={{background:'#f59e0b'}}>Контролерів онлайн: <strong
style={{marginLeft:8}}>6</strong></div>
      </div>

      <div className="card" style={{marginBottom:12}}>
        <h3>Активність відвідувань за тиждень</h3>
        <div style={{width:'100%', height:220}}>
          <ResponsiveContainer width="100%" height="100%">
            <AreaChart data={data}>
              <defs>
                <linearGradient id="colorPv" x1="0" y1="0" x2="0" y2="1">
```

```
              <stop offset="5%" stopColor="#0369a1" stopOpacity={0.8}/>
              <stop offset="95%" stopColor="#0369a1" stopOpacity={0}/>
            </linearGradient>
          </defs>
          <XAxis dataKey="day" />
          <Tooltip />
          <Area type="monotone" dataKey="visits" stroke="#0369a1" fillOpacity={1}
fill="url(#colorPv)" />
        </AreaChart>
      </ResponsiveContainer>
    </div>
  </div>

  <div style={{display:'flex', gap:12}}>
    <div className="card" style={{flex:1}}>
      <h3>Останні події</h3>
      <ul style={{listStyle:'none', padding:0}}>
        <li style={{padding:8, borderTop:'1px solid #eef2f6'}}>Ірина Петренко — Доступ
дозволено — Тренажерний зал</li>
        <li style={{padding:8, borderTop:'1px solid #eef2f6'}}>Анна Петренко — Доступ
заборонено — Басейн</li>
        <li style={{padding:8, borderTop:'1px solid #eef2f6'}}>Олег Кравчук — Відкрив
двері — Вхід №1</li>
      </ul>
    </div>
    <div className="card" style={{width:300}}>
      <h3>Статус системи</h3>
      <div>API: <strong style={{color:'#065f46'}}>Онлайн</strong></div>
      <div>База даних: <strong style={{color:'#065f46'}}>Онлайн</strong></div>
      <div>Логування: <strong>Увімкнено</strong></div>
    </div>
  </div>
 );
}
```

Лістинг К.2 — Лістинг файлу LogsPage.jsx

```
import React, { useState } from 'react';
export default function LogsPage(){
```

```
  const [q, setQ] = useState('');
  const mock = [
    {id:1, time:'2025-11-04 10:05', user:'Ірина Мельник', zone:'Тренажерний зал',
action:'Доступ дозволено'},
    {id:2, time:'2025-11-04 10:15', user:'Анна Петренко', zone:'Басейн', action:'Доступ
заборонено'},
    {id:3, time:'2025-11-04 11:00', user:'Олег Кравчук', zone:'Вхід №1', action:'Відкрив
двері'}
  ];
  const filtered = mock.filter(l=> l.user.toLowerCase().includes(q.toLowerCase()) ||
l.zone.toLowerCase().includes(q.toLowerCase()));
  const exportCSV = ()=>{
    const rows = [['Час','Користувач','Зона','Подія'],
...filtered.map(r=>[r.time,r.user,r.zone,r.action])];
    const csv = rows.map(r=>r.map(c=>`"${String(c).replace(/"/g,'""')}"`).join(',')).join('\n');
    const blob = new Blob([csv], {type:'text/csv'});
    const url = URL.createObjectURL(blob);
    const a = document.createElement('a'); a.href=url; a.download='logs.csv'; a.click();
URL.revokeObjectURL(url);
  };
  return (
    <div className="container" style={{paddingTop:20}}>
      <div style={{display:'flex', justifyContent:'space-between', marginBottom:12}}>
        <h2>Журнал подій</h2>
        <div>
          <input placeholder="Пошук по користувачу або зоні" value={q}
onChange={e=>setQ(e.target.value)} />
          <button onClick={exportCSV} style={{marginLeft:8}}>Експорт CSV</button>
        </div>
      </div>
      <div className="card">
        <table className="table">

<thead><tr><th>Час</th><th>Користувач</th><th>Зона</th><th>Подія</th></tr></thead
>
          <tbody>
            {filtered.map(r=>(
              <tr
key={r.id}><td>{r.time}</td><td>{r.user}</td><td>{r.zone}</td><td>{r.action}</td></tr>
```

```
          ))}
        </tbody>
      </table>
    </div>
  </div>
 );
}
```

Лістинг К.3 — Лістинг файлу UsersPage.jsx

```
import React, { useEffect, useState } from 'react';
export default function UsersPage(){
 const [q, setQ] = useState('');
 const [users, setUsers] = useState([]);
 const [showModal, setShowModal] = useState(false);
 const [editing, setEditing] = useState(null);
 const mock = [
  {id:1, name:'Олег Кравчук', role:'Адміністратор', rfid:'A1B2C3'},
  {id:2, name:'Ірина Мельник', role:'Користувач', rfid:'D4E5F6'},
  {id:3, name:'Анна Петренко', role:'Тренер', rfid:'Z9Y8X7'}
 ];
 useEffect(()=>{ setUsers(mock); }, []);
 const filtered = users.filter(u=> u.name.toLowerCase().includes(q.toLowerCase()) ||
u.role.toLowerCase().includes(q.toLowerCase()));
 const save = (user)=>{ if(user.id){ setUsers(users.map(u=>u.id===user.id?user:u)); } else {
user.id = Date.now(); setUsers([user,...users]); } setShowModal(false); };
 return (
  <div className="container" style={{paddingTop:20}}>
   <div style={{display:'flex', justifyContent:'space-between', marginBottom:12}}>
    <h2>Користувачі</h2>
    <div>
     <input placeholder="Пошук за ім'ям або роллю" value={q}
onChange={e=>setQ(e.target.value)} />
     <button onClick={()=>{ setEditing(null); setShowModal(true); }}
style={{marginLeft:8}}>Додати</button>
    </div>
   </div>
   <div className="card">
    <table className="table">
```

```
<thead><tr><th>ID</th><th>ПІБ</th><th>Роль</th><th>RFID</th><th></th></tr></thead>
        <tbody>
          {filtered.map(u=>(
            <tr key={u.id}>
              <td>{u.id}</td><td>{u.name}</td><td>{u.role}</td><td>{u.rfid}</td>
              <td><button onClick={()=>{ setEditing(u); setShowModal(true); }}>Редагувати</button></td>
            </tr>
          ))}
        </tbody>
      </table>
    </div>

    {showModal && <UserModal onClose={()=>setShowModal(false)} onSave={save} user={editing} />}
  </div>
 );
}

function UserModal({onClose, onSave, user}){
 const [name, setName] = useState(user?.name || '');
 const [role, setRole] = useState(user?.role || 'Користувач');
 const [rfid, setRfid] = useState(user?.rfid || '');
 return (
  <div style={{position:'fixed', inset:0, display:'flex', alignItems:'center', justifyContent:'center', background:'rgba(2,6,23,0.4)'}}>
    <div style={{width:420}} className="card">
      <h3>{user ? 'Редагувати користувача' : 'Додати користувача'}</h3>
      <div style={{marginTop:8}}>
        <label>ПІБ</label><br/>
        <input value={name} onChange={e=>setName(e.target.value)} />
      </div>
      <div style={{marginTop:8}}>
        <label>Роль</label><br/>
        <select value={role} onChange={e=>setRole(e.target.value)}>

<option>Користувач</option><option>Тренер</option><option>Адміністратор</option>
```

```
          </select>
        </div>
        <div style={{marginTop:8}}>
          <label>RFID</label><br/>
          <input value={rfid} onChange={e=>setRfid(e.target.value)} />
        </div>
        <div style={{marginTop:12, display:'flex', justifyContent:'flex-end', gap:8}}>
          <button onClick={onClose}>Скасувати</button>
          <button onClick={()=>onSave({id:user?.id, name, role, rfid})}>Зберегти</button>
        </div>
      </div>
    </div>
  );
}
```