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

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Комп'ютерна онлайн система обліку, рейтингування та контролю проживання студентів у гуртожитку»

Виконав: студент 2 курсу, групи 1КІ-23мс
спеціальності 123 — «Комп'ютерна інженерія»
освітня програма – «Комп'ютерна інженерія»
Маліцький В.В.
Керівник: к.т.н., проф. каф. ОТ
Войцеховська О. В.
Рецензент: к.т.н., доц., доцент каф. ПЗ
Чехместрук Р. Ю.

Допущено до захисту
Завідувач кафедри ОТ
д.т.н., проф. Азаров О. Д.

«____» _____ 2025 р.

Вінниця ВНТУ – 2025 рік

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Освітньо—кваліфікаційний рівень бакалвр
Галузь знань — 12 Інформаційні технології
Спеціальність — 123 Комп'ютерна інженерія
Освітня програма — Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
д.т.н.проф._____Азаров О. Д.
«21» березня 2025 року

З А В Д А Н Н Я

НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Маліцькому Владиславу Віталійовичу

1 Тема роботи: «Комп'ютерна онлайн система обліку, рейтингування та контролю проживання студентів у гуртожитку», керівник роботи Войцеховська О.В. к.т.н., доц., доцент кафедри ОТ затверджені наказом вищого навчального закладу від «20» березня 2025 року № 97.

2 Термін подання студентом роботи 13.05.2025

3 Вихідні дані до роботи: план гуртожитка; особисті дані студентів; акти порушень; документи для рейтингування; сучасні технології розробки клієнтської та серверної частин, середовище розробки — Visual Studio Code.

4 Зміст розрахунково-пояснювальної записки Вступ. Аналіз сучасного стану розвитку галузі розробки онлайн систем контролю проживання студентів у гуртожитках. Проєктування онлайн системи контролю проживання студентів у гуртожитках. Програмна реалізація онлайн системи контролю проживання студентів у гуртожитках. Програмна реалізація онлайн системи контролю проживання студентів у гуртожитках.

5 Перелік графічного матеріалу: Структурна схема архітектурних рівнів

системи, ER-діаграма бази даних, Структурна схема клієнтської части Структурна схема серверної частини. Перелік ілюстраційного матері Скриншоти графічного інтерфейсу системи.

6 Консультанти розділів роботи приведені в таблиці 1.

Таблиця 1 — Консультанти роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1 - 4	к.т.н., доц., доцент кафедри ОТ Войцеховська О.В.	21.03.25	13.05.25
Нормоконтроль	ас. каф. ОТ Швець С. І.	14.05.25	30.05.25

7 Дата видачі завдання 21.03.2025

8 Календарний план приведений в таблиці 2.

Таблиця 2 — Календарний план

№ з/п	Назва етапів дипломної роботи	Строк виконання	Підпис
1.	Постановка задачі роботи	21.03.25	виконано
2.	Аналіз сучасного стану розвитку галузі розробки онлайн систем контролю проживання студентів у гуртожитках	21.03.25—30.03.25	виконано
3.	Проєктування онлайн системи контролю проживання студентів у гуртожитках	21.03.25—30.03.25	виконано
4.	Програмна реалізація онлайн системи	31.03.25—13.04.25	виконано
5.	Тестування онлайн системи	14.04.25—20.04.25	виконано
6.	Впровадження онлайн системи контролю проживання студентів у гуртожитках	21.04.25—27.04.25	виконано
7.	Оформлення пояснювальної записки та ілюстративного матеріалу	28.04.25—11.05.25	виконано
8.	Перевірка якості виконання бакалаврської кваліфікаційної роботи та усунення недоліків	13.05.2025	виконано

Студент Владислав МАЛІЦЬКИЙ

Керівник роботи к.т.н., доц. каф. ОТ Олена ВОЙЦЕХОВСЬК

АНОТАЦІЯ

УДК 004.42

Маліцький В. В. Онлайн система обліку, рейтингування та контролю проживання студентів у гуртожитку. Бакалаврська кваліфікаційна робота (проєкт) зі спеціальності 123 «Комп'ютерна інженерія», освітня програма «Комп'ютерна інженерія». Вінниця: ВНТУ, 2022. 83 с.

На укр. мові. Бібліогр.: 39 назв; рис.: 34; табл. 13.

У бакалаврській кваліфікаційній роботі розроблено онлайн систему обліку, рейтингування та контролю проживання студентів у гуртожитку, яка дає можливість покращити документообіг в гуртожитках, спростити процес рейтингування та контролю за дотриманням правил проживання. Проведено порівняльний аналіз аналогів існуючих систем. Проаналізовано сучасні технології для розробки вебзастосунків для подібних систем. Розроблено структуру та архітектуру системи, а також вебзастосунок, що дає можливість вести облік студентів, рейтингування, контролювати поселення та виселення, слідкувати за порушеннями.

Ключові слова: комп'ютерна система, React, MySQL, Firebase, Node.js облік студентів, рейтингування, контроль проживання, гуртожиток.

ABSTRACT

Malitskyi V. V. Online system of accounting, rating and control of student accommodation in a hostel. Bachelor's qualification work (project) in specialty 123 ‘Computer Engineering’, educational programme ‘Computer Engineering’. Vinnytsia: VNTU, 2022. 83 p.

In Ukrainian language. Bibliographer: 39 titles; Fig.: 34; table 13.

In the bachelor's qualification work, an online system of accounting, rating and control of student accommodation in a dormitory was developed, which makes it possible to improve the document flow in dormitories, simplify the process of rating and monitoring compliance with the rules of residence in a dormitory. A comparative analysis of analogues of existing systems is carried out. Modern technologies for developing web applications for such systems are analysed. The structure and architecture of the system, as well as a web application that allows keeping records of students, rating, controlling settlements and evictions, and monitoring violations, are developed.

Keywords: computer system, React, MySQL, Firebase, Node.js, student accounting, rating, accommodation control, dormitory.

ЗМІСТ

ВСТУП

У сучасному світі, де цифровізації освітніх процесів приділяють особливу увагу, важливим є використанням нових технологій. Однією з ключових сфер, яка потребує систематизації та оптимізації, є управління проживанням студентів у гуртожитках. Більшість вищих навчальних закладів досі використовують ручні або застарілі методи обліку мешканців, ведення документації та контролю дисциплінарних порушень, що призводить до неефективності, затримок в обробці інформації та зниження прозорості. Також, це створює об'єктивну потребу у створенні комп’ютерної онлайн системи, яка б дозволяла централізовано керувати всіма аспектами проживання студентів.

Метою кваліфікаційної роботи є розширення функціональних можливостей наявної системи обліку для гуртожитку шляхом створення комп’ютерної онлайн системи, яка забезпечить можливість формування рейтингу, контролю дисциплінарних заходів, ведення звітності за окремим студентом та збереження супровідних документів.

Задачі, які необхідно виконати для досягнення поставленої мети:

— проаналізувати архітектурні підходи до написання розподілених веб-додатків;

—провести аналіз існуючих технологій розробки серверної частини веб-додатків та визначити необхідний стек технологій для розробки;

—провести аналіз існуючих технологій розробки клієнтської частини веб-додатків та визначити необхідний стек технологій для розробки;

— провести порівняльний аналіз існуючих аналогів онлайн систем контролю проживання студентів у гуртожитках;

— здійснити проектування архітектури онлайн системи;

— реалізувати клієнтську частину веб-додатку;

—реалізувати серверну частину веб-додатку;

— реалізувати базу даних для онлайн системи контролю проживання студентів у гуртожитках;

—здійснити тестування клієнтської частини

Об’єктом дослідження є процеси управління проживанням студентів у гуртожитках вищих навчальних закладів.

Предметом дослідження є засоби, методи та технології створення інформаційних комп’ютерних онлайн систем.

У межах дослідження було реалізовано систему, яка об'єднує функції реєстрації студентів, поселення, переселення, фіксації порушень і заслуг, автоматичного формування рейтингу та звітності з інтерактивною мапою гуртожитку та централізованим зберіганням документів через Google Drive. Рішення адаптоване до потреб українських освітніх закладів і може масштабуватися для використання в інших установах.

Апробація результатів кваліфікаційної роботи: зроблено доповідь на LIV Всеукраїнській науково-технічній конференції підрозділів Вінницького національного технічного університету (2025).

Публікації:

1. Онлайн система обліку, рейтингування та контролю проживання студентів у гуртожитку // Маліцький В. В., Войцеховська О. В. Матеріали LIV Всеукраїнської науково-технічної конференції підрозділів Вінницького національного технічного університету (Вінниця, 2025 р.) [Електронний ресурс]. Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2025/paper/view/23812/19633 [1].

2. Свідоцтво про реєстрацію авторського права на твір № 135686. Комп’ютерна програма «Програмний модуль реєстрації порушень правил проживання студентів у гуртожитку» / Бабюк Н. П., Войцеховська О. В., Маліцький В. В. Дата реєстрації 05.05.2025 р. [2].

1 АНАЛІЗ СУЧАСНОГО СТАНУ РОЗВИТКУ ГАЛУЗІ РОЗРОБКИ ОНЛАЙН СИСТЕМ КОНТРОЛЮ ПРОЖИВАННЯ СТУДЕНТІВ У ГУРТОЖИТКАХ

1.1 Порівняльний аналіз аналогів онлайн систем контролю проживання студентів у гуртожитках

На даний час, існують загальні системи управління навчальним процесом, наприклад CRM-системи, АСУ навчального закладу або ведення документообігу через Excel. Проте саме спеціалізованих рішень, які повністю охоплювали б управління гуртожитками, облік поселення, рейтингування студентів, контроль порушень та зручну візуалізацію розміщення, практично не існує. Розглянемо декілька з них [3].

Автоматизовані системи управління навчальним закладом, скорочено АСУ зазвичай включають модулі для обліку студентів, реєстрації, відвідуваності, виставлення оцінок та формування звітності. Таким прикладом є платформна JetIQ від Вінницького національно технічного університету та АСУ «ВНЗ» від Наукового-дослідницького інституту Прикладних інформаційних технологій. В АСУ «ВНЗ» пакет «Деканат» є досить поширеним у державних вищих навчальних закладках України, понад 70 навчальних закладів України вже успішно використовують систему.

АСУ «ВНЗ» виконує велику кількість завдань освітнього закладу. Складається з таких компонентів як, АС «Приймальна комісія», АС «Деканат», АС «Студмістечко» та АС «Конструктор звітів». Перелік компонентів АСУ «ВНЗ» наведено на рисунку 1.1 [4].

Розглянемо, автоматизовану систему «Студмістечко» — це програмно-технологічний комплекс, який є складовою частиною автоматизованої системи керування «Деканат+» і призначений для автоматизації всіх процесів, починаючи від розподілу місць в гуртожитках та контролю оплат до управління пропускною системою [5].

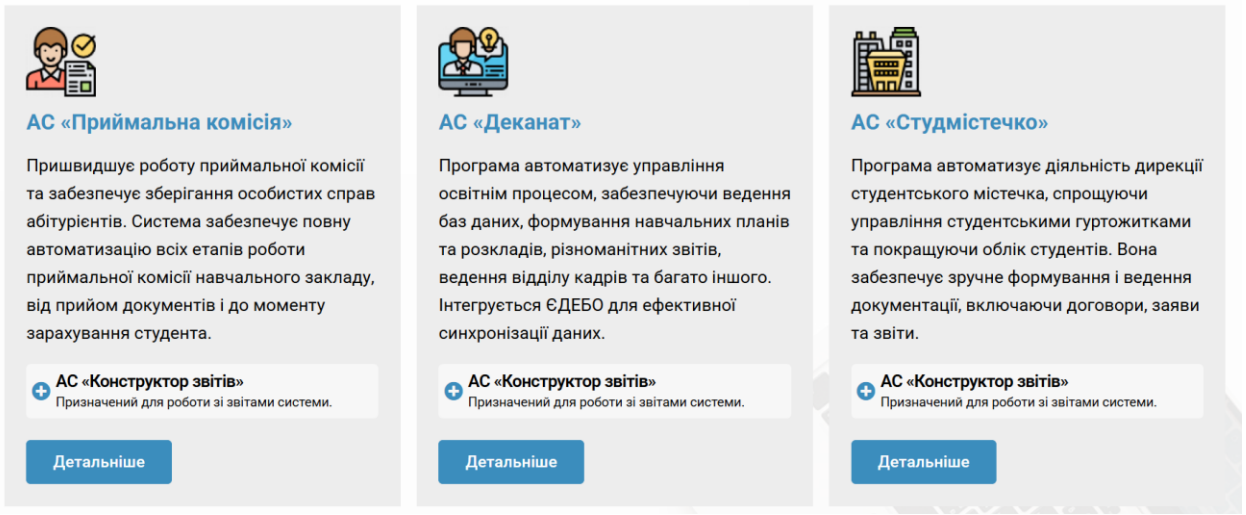

Рисунок 1.1 — Перелік компонентів АСУ «ВНЗ»

Основними функціями системи «Студмістечко» є поселення студентів на основі даних із АС «Деканат», автоматизація діяльності дирекції студентського містечка навчального закладу, спрощення та вдосконалення документообігу, підвищення оперативності, узгодженості та достовірності даних. Також система передбачає автоматизацію всіх процесів — від розподілу місць і контролю до формування звітів та управління пропускною системою. Крім того, вона забезпечує можливість друку поточної, службової, статистичної та звітної документації відповідно до внесених даних.

Водночас система має низку обмежень: вона орієнтована виключно на навчальний процес, не підтримує візуалізацію розміщення студентів у гуртожитку, відзначається недостатньою гнучкістю для створення рейтингів чи фіксації порушень. Крім того, система має закриту архітектуру, що ускладнює розширення функціональних можливостей, а також потребує витрат на підтримку.

JetIQ являє собою багатомодульну інформаційну платформу, яка активно використовується у Вінницькому національному технічному університеті. Основна мета JetIQ — об’єднання всіх ключових процесів в межах університету: від навчального обліку до фінансових операцій, включно з базовим модулем обліку проживання студентів у гуртожитках [6].

Один із модулів, пов’язаний з обліком проживання студентів у гуртожитках, — модуль JetIQ, який відповідає за гуртожитки. У системі передбачено можливість

подання заяв на поселення від студентів та подальше підтвердження заяв адміністрацією університету. Вигляд таблиці «Заяви на поселення» наведено на рисунку 1.2.

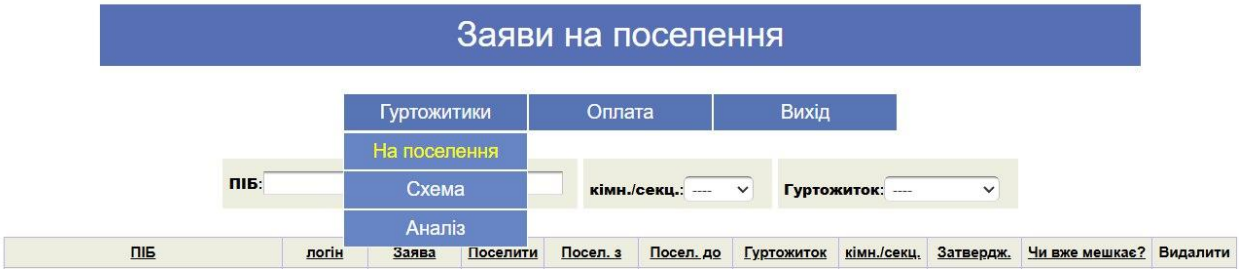

Рисунок 1.2 — Вигляд таблиці «Заяви на поселення» в JetIQ

Поточний стан заповненості кімнат можливо переглянути за допомогою кольорової схеми: зелений (заселено повністю), жовтий (частково заселено), червоний (переповнено). При наведенні на потрібну кімнату, відображається список студентів, які там проживають. Схема розселення у гуртожитках наведено на рисунку 1.2.

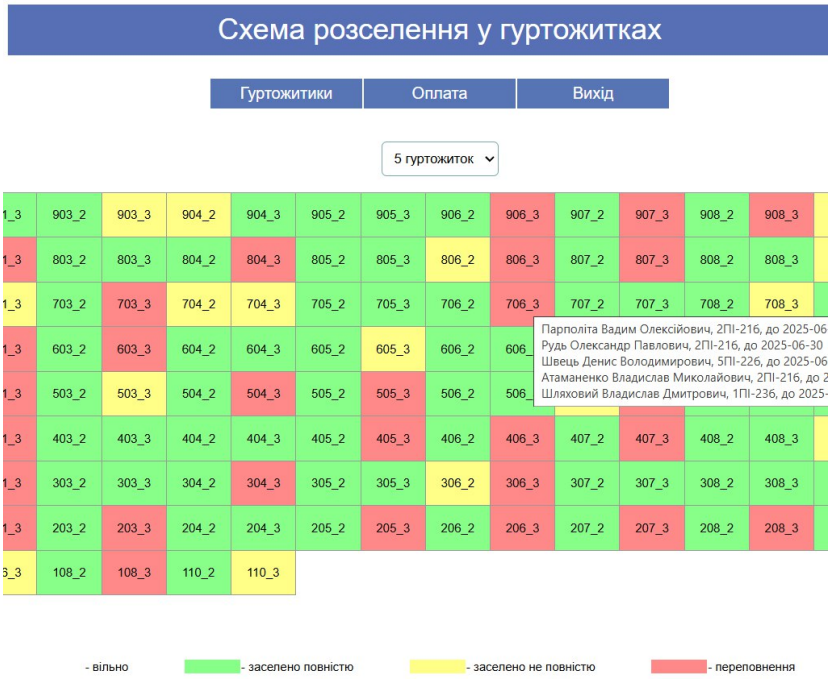

Рисунок 1.3 — Схема розселення у гуртожитках в JetIQ

Також, система надає можливість здійснювати фільтрацію даних за секціями, номерами кімнат і прізвищами студентів. Це дозволяє адміністративному

персоналу здійснювати точний пошук і проводити аналіз розміщення. Вигляд вікна аналізу розселення наведено на рисунку 1.4.

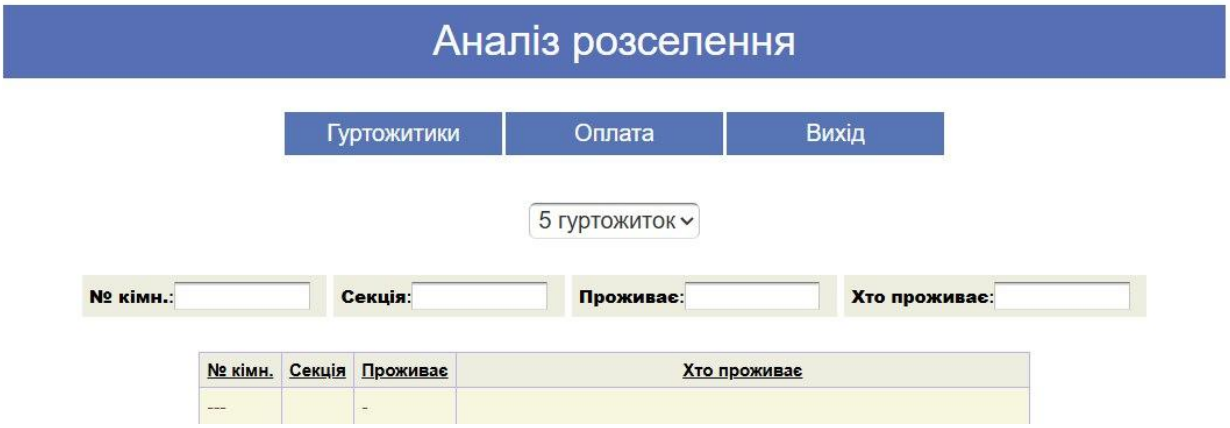

Рисунок 1.4 — Вигляд вікна аналізу розселення в JetIQ

Крім того, в цьому модулі реалізовано функцію перегляду інформації про оплату за проживання студентів.

Також, багато навчальних закладів користуються простими електронними таблицями, наприклад, Google Sheets або Excel, щоб вести списки студентів, фіксувати поселення та інші базові дані. Це найпростіший та найшвидший спосіб організації обліку [7].

Перевагами такого підходу є безкоштовність і простота у використанні, гнучкість у побудові формул і розрахунків, а також можливість колективного редагування даних у Google Sheets. Водночас цей метод має суттєві обмеження, наприклад як відсутність контролю доступу за ролями, низький рівень структурованості даних, а також непридатність для реалізації складної логіки, таку як, облік актів, автоматичного рейтингування студентів чи побудови інтерактивних мап. Крім того, у таких системах відсутня автоматизація процесів, а дані можуть легко пошкоджуватися або видалятися випадково.

Отже, електроні таблиці — це тимчасове рішення, що не масштабується та не забезпечує безпеку й автоматизацію.

Наведені аналоги демонструють, що жодна з наявних систем не надає повноцінного, спеціалізованого набору функцій для ефективного управління гуртожитками.

Було проведено аналіз сучасного стану проблеми обліку та контролю проживання студентів у гуртожитках закладів вищої освіти. Встановлено, що в більшості випадків адміністрація гуртожитків та адміністрація університетів використовують застарілі або неуніфіковані інструменти управління, які не забезпечують прозорості, оперативності та зручності в роботі з великою кількістю студентів і житлових одиниць.

Виконано аналіз існуючих рішень, зокрема таких систем як JetIQ, АСУ «Деканат+», Google Таблиці та CRM-платформи, показав їхню непридатність або обмежену ефективність для вирішення завдань, пов’язаних із житловим обліком. Жоден з існуючих аналогів не надає повного набору функціональних можливостей для візуального управління гуртожитками, рейтингування мешканців, формування актів порушень та взаємодії між відповідальними особами та деканатом.

Отже, розроблювана система повинна розширити можливості JetIQ, орієнтуючись саме на потреби деканату вищого навчально закладу та адміністрації гуртожитків. Також необхідно враховувати взаємодію між кількома гуртожитками, потребу в інтерактивній мапі, гнучкому рейтингу та процедурі затвердження актів порушень. Така система дозволяє забезпечити прозорість, масштабованість та контроль.

1.2 Постановка задачі дослідження онлайн системи контролю проживання студентів у гуртожитках

Система реалізується у вигляді вебзастосунку, що дозволяє здійснювати централізований облік студентів, контролювати їхнє проживання, фіксувати порушення, формувати рейтинг та працювати з інтерактивною мапою гуртожитку.

Передбачено створення інтерфейсу для:

— додавання нових студентів до системи;

— редагування анкетних даних студентів;

— закріплення за певною кімнатою у гуртожитку;

— фіксації переселеннння і виселення із журналом змін;

— перегляду поточного розміщення студентів.

Потрібно реалізувати графічне представлення поверхів та кімнат гуртожитку. Інтерактивна мапа гуртожитку повинна дозволяти:

— бачити візуально зайняті та вільні кімнати;

— переглядати інформацію про мешканців;

— обирати кімнати при поселенні;

— швидко орієнтуватися у структурі гуртожитку.

Система повинна дозволяти одночасно вести облік по кількох гуртожитках. Користувач з відповідними правами зможе:

— переглядати список доступних гуртожитків;

— перемикатися між ними.

Передбачається створення механізму формування рейтингів мешканців на основі:

— наявності та кількості порушень;

— заохочень, участі в ініціативах гуртожитку.

Рейтинг відображатиметься числовим значенням та буде використовуватися при формуванні звітів і прийнятті адміністративних рішень.

У відповідальних осіб має бути можливість запропонувати акт порушення або заохочення, в якому зазначатимуться такі дані:

— студент;

— дата;

— опис звернення;

— файл з описом порушення.

Акт автоматично надсилатиметься на розгляд адміністратору системи, який зможе затвердити або відхилити його.

Також має бути реалізований модуль звітності, що включатиме:

— рейтингові таблиці по гуртожитках;

— історію порушень.

Дані повинні бути доступні для експорту у форматах CSV або Excel.

У результаті створення та впровадження системи очікується наступне:

— централізований облік студентів у гуртожитках;

— оперативне реагування на порушення через модуль актів;

— зручний інтерфейс для деканів і відповідальних осіб;

— впровадження прозорої системи рейтингування для прийняття рішень;

— зменшення паперового документообігу;

— можливість масштабування при додаванні нових гуртожитків.

У результаті аналізу проблеми та постановки завдання було визначено комплекс функціональних модулів, які необхідно реалізувати в межах комп'ютерної онлайн системи обліку, рейтингування та контролю проживання студентів у гуртожитках. Запропоноване рішення передбачає створення веборієнтованої платформи з чітким розподілом ролей користувачів, інтерактивною мапою гуртожитків, системою фіксації порушень, рейтингуванням студентів, а також аналітичними засобами для деканату та адміністрації.

2 ПРОЄКТУВАННЯ ОНЛАЙН СИСТЕМИ КОНТРОЛЮ ПРОЖИВАННЯ СТУДЕНТІВ У ГУРТОЖИТКАХ

Враховуючи проблеми обліку проживання студентів у гуртожитках, запропоновано розширити функціональні можливості наявної у ВНТУ системи обліку для гуртожитку (модуль в JetIQ) шляхом створення комп'ютерної онлайн системи, яка забезпечить можливість формування рейтингу, контролю дисциплінарних заходів, ведення звітності за окремим студентом та збереження супровідних документів, орієнтовану на потреби адміністрації гуртожитку, деканату та відповідальних осіб.

2.1 Обґрунтування технічних вимог до онлайн систем контролю проживання студентів у гуртожитках

Для повноцінного функціонування розроблюваної комп'ютерної онлайн системи необхідно сформулювати чіткі технічні вимоги. Вони охоплюють як функціональні можливості, які має реалізовувати система, так і нефункціональні обмеження та критерії якості, що забезпечують стабільну, зручну і безпечну роботу в умовах реального навчального закладу.

Запропонована система повинна підтримувати багаторівневу авторизацію, тобто роль представника деканата, відповідальної особи та адміністратора системи.

Система повинен надавати функції додавання, редагування, видалення студентів, прив'язки студента до кімнати та фіксація історії поселень, переселень, виселень.

Система повинна надавати можливість додавання кількох гуртожитків та прив'язувати поверхи і кімнати до конкретного гуртожитку.

Система повинна надавати дозвіл до додавання порушень або заохочень відповідальними особами, формування акту із зазначенням деталей, відправка акту на затвердження декану, а також можливість для представника деканату підтвердити або відхилити акт.

Система повинна надавати можливість створення аналітичних звітів, фільтрації за гуртожитком, кімнатою, рейтингом, факультетом та експорт таблиць у формат Excel.

Система повинна бути доступною 24/7 через вебінтерфейс та мати підтримку доступу з персональний комп'ютери, планшетів, смартфонів.

Відгук системи на будь-яку дію не повинен перевищувати 1 секунди при навантаженні до 50 одночасних користувачів.

Система повинна підтримувати додавання нових гуртожитків без зміни архітектури. А також мати можливість подальшого розширення, наприклад впровадження мобільного застосунку, API тощо.

Клієнтська частина має працювати в сучасних браузерах (Google Chrome, Mozilla Firefox, Microsoft Edge), бути розроблена з використанням HTML5, CSS3, JavaScript. Також має бути реалізована адаптивна верстка для коректного відображення на різних пристроях.

Серверна частина повинна реалізована на сучасних технологіях, підтримувати REST API для взаємодії з клієнтом та мати підключення до реляційної бази даних.

База даних повинна відповідати за зберігання інформації про студентів, гуртожитки, кімнати, дії, рейтинги, акти та забезпечення цілісності, резервного копіювання та відновлення.

Інфраструктура системи повинна мати хостинг на сервері з підтримкою HTTPS, підтримку автоматичних оновлень системи.

Визначені технічні вимоги охоплюють усі ключові аспекти створення надійної, безпечної та масштабованої системи обліку проживання студентів у гуртожитках. Їх дотримання гарантуватиме ефективну роботу платформи в умовах реального використання адміністрацією навчального закладу.

2.3 Вибір архітектури онлайн системи контролю проживання студентів у гуртожитках

Одним з головних етапів розробки програмного забезпечення є вибір

архітектурного підходу, що визначає структуру, принципи взаємодії компонентів системи, розподіл функцій між ними та можливості масштабування [7].

Монолітна модель передбачає об’єднання всієї бізнес-логіки, інтерфейсу та доступу до даних у межах єдиного застосунку. Подібний підхід спрощує початкову реалізацію та розгортання, але ускладнює масштабування та супровід системи. У разі зростання кількості користувачів чи необхідності впровадження нових функцій доведеться змінювати увесь застосунок, що спричиняє ризики помилок і конфліктів [8].

Мікросервісна архітектура базується на розподілі системи на незалежні модулі, кожен з яких відповідає за окрему функцію. Такий підхід забезпечує високу масштабованість, гнучкість та знаходження помилок саме в модулях. Проте він є складнішим у реалізації, потребує незалежного розгортання сервісів, взаємодії через API. Для освітньої системи локального масштабу мікросервіси можуть бути надлишковими з точки зору витрат ресурсів.

Трирівнева модель структурує застосунок на інтрефейсний рівень, логічний рівень та рівень даних. Це дозволяє гнучко відокремити відповідальність і створювати незалежні компоненти. Така архітектура часто використовується в корпоративних системах, однак її реалізація є тісно пов’язаною з клієнт-серверною моделлю. Схему взаємодії компонентів клієнт-серверного вебзастосунку наведено на рисунку 2.1.

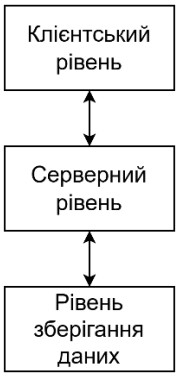

Рисунок 2.1 — Схема взаємодії компонентів клієнт-серверного вебзастосунку

Для реалізації онлайн системи обліку, рейтингування та контролю проживання студентів у гуртожитках обрано архітектуру клієнт-серверного

вебзастосунку, що дозволяє забезпечити доступність, масштабованість та зручність використання.

2.4. Вибір технології розробки клієнтської частини системи контролю проживання студентів у гуртожитках

У процесі розробки інтерфейсу користувача важливим етапом є визначення оптимального інструменту для побудови клієнтської частини системи. Серед найбільш популярних рішень виділяються три технології: React, Angular та Vue. Кожна з них має власну архітектуру, підходи до побудови інтерфейсу та рівень складності реалізації, тому потребує окремого аналізу.

Angular — це повноцінний фреймворк, створений компанією Google. На відміну від React, Angular має чітку структуру, вбудовану підтримку для роботи з формами, HTTP-запитами, маршрутизацією, реактивним програмуванням через RxJS і TypeScript за замовчуванням. Angular добре підходить для великих корпоративних проектів з великою кількістю модулів, де важливо дотримуватися суворої архітектури [9].

Водночас, освоєння Angular вимагає глибшого технічного розуміння. Розробнику необхідно опанувати роботу з декораторами, ін’єкцією залежностей, модулями, шаблонами, сервісами, а також налаштуванням складного конфігураційного середовища. Це зумовлює більший обсяг навчального матеріалу та практики, ніж у інших рішеннях. Крім цього, реалізація проектів в Angular часто вимагає більше часу через суворі структурні вимоги та кількість обов’язкових компонентів для побудови простих елементів керування.

React — це сучасна JavaScript-бібліотека з відкритим кодом, розроблена компанією Meta, що була раніше компанією Facebook, спільно з незалежними розробниками, яка призначена для створення інтерфейсів користувача. Бібліотека вирішує задачу часткового оновлення контенту вебсторінки, що особливо актуально для односторінкових застосунків [10].

React надає можливість створювати масштабовані та продуктивні вебзастосунки, де дані змінюються динамічно без потреби перезавантаження

сторінки. Основна можливість бібліотеки — це побудова інтерфейсу користувача. React легко інтегрується з іншими бібліотеками або фреймворками, а також може бути доповнений надбудовами для управління логікою поза межами інтерфейсу.

Vue.js — це прогресивний фреймворк, який займає проміжне положення між React і Angular. Він поєднує легкість засвоєння React з архітектурними можливостями Angular, пропонуючи зрозумілий API, підтримку реактивності, компонентний підхід і вбудовану маршрутизацію через Vue Router. Vue має компактне ядро та багатий набір офіційних розширень, однак глобальна спільнота та корпоративна підтримка є дещо меншою, ніж у React або Angular [11].

Для реалізації інтерфейсу користувача розроблюваної системи контролю проживання студентів у гуртожитках обрано технологію React. Компонентний підхід React дозволяє створювати незалежні багаторазові компоненти інтерфейсу, такі як картка студента, мапа кімнат, таблиці для даних за фільтрами. Також React легко підтримує зростання кількості функцій і ускладнення логіки.

У межах розробки онлайн системи контролю проживання студентів у гуртожитках за допомогою React буде реалізовано: форму авторизації та панель користувача; перегляд і редагування списків студентів; інтерактивну мапа гуртожитку; таблиці рейтингів, фільтрація, пошук; роботу з актами порушень; розмежування доступу за ролями.

2.5 Вибір технології розробки серверної частини онлайн системи контролю проживання студентів у гуртожитках

Технологій для розробки серверної частини будь-якої сучасної інформаційної системи повинні забезпечувати стабільну, швидку та безпечну обробку запитів, взаємодію з базою даних, підтримку авторизації та можливість масштабування в майбутньому. Сервер є ключовою ланкою у моделі клієнт-серверної архітектури, оскільки саме на ньому зосереджена вся бізнес-логіка, обробка запитів, керування ролями та зв’язок із базою даних і зовнішніми сервісами.

Фреймворк Django — це високорівневий фреймворк з відкритим вихідним кодом, створений на мові програмування Python для розробки вебдодатків [12].

Однією з ключових особливостей Django є модульна структура, тобто вебзастосунок складається з однієї або кількох частин, які розробляються як окремі модулі. Такий підхід вирізняє Django серед інших фреймворків. Проте він базується на мові Python, яка є менш сумісною з фронтендом, побудованим на JavaScript. Подібне рішення ускладнює взаємодію між клієнтом і сервером у SPA-застосунках, де важлива уніфікація мов та типів даних.

Фреймворк Laravel — це безкоштовний PHP-фреймворк з відкритим вихідним кодом, розроблений Тейлором Отвелом для створення вебдодатків на основі архітектурної моделі MVC [13].

Серед ключових особливостей Laravel є модульна система з використанням менеджера залежностей Composer, підтримка кількох способів взаємодії з реляційними базами даних, набір утиліт для розгортання та обслуговування додатків, а також акцент на зручний і читабельний синтаксис

Проте PHP, як мова, вважається менш придатною для асинхронних операцій, а Laravel орієнтований більше на монолітні або трьох компонентні застосунки, ніж на сучасні SPA з окремим фронтендом.

Node.js — це відкрита платформа для створення високопродуктивних мережевих застосунків мовою JavaScript. На відміну від звичайного використання JavaScript лише в браузерах, Node.js дозволяє виконувати JavaScript-код на серверній стороні, завдяки чому JavaScript став універсальною мовою програмування з активною спільнотою [14].

Основні характеристики Node.js полягають у роботі за асинхронною однонитковою моделлю обробки запитів і використанні неблокуючого введення/виведення. Node.js базується на високопродуктивному рушії Google V8, застосовує систему модулів CommonJS та має вбудовану підтримку менеджера пакунків Node Package Manager (npm), що значно спрощує керування залежностями у проєкті.

Npm — це стандартний пакетний менеджер для Node.js, який забезпечує управління бібліотеками та залежностями. Він складається з клієнта, командного рядка і реєстру. Реєстр являє собою онлайн-сховище відкритих і приватних пакунків. Через цей реєстр розробники можуть легко знаходити, встановлювати та публікувати пакунки [15].

Для реалізації серверної частини розроблюваної системи контролю проживання студентів у гуртожитках обрано Node.js.

У рамках реалізації серверної частини вебзастосунку буде використано Express.js — легкий і гнучкий фреймворк для розробки серверної частини вебзастосунків на платформі Node.js. Він є вільним програмним забезпеченням з відкритим кодом, розповсюджується під ліцензією MIT і слугує одним із найпопулярніших інструментів для створення вебсервісів та API [16].

Express вважається стандартом серед фреймворків для Node.js. Express є мінімалістичним за структурою, проте підтримує велику кількість розширень і модулів.

Перевагою використання Node.js разом з Express є можливість реалізації як клієнтської, так і серверної частини застосунку за допомогою однієї мови програмування — JavaScript. Швидкодія завдяки асинхронній моделі введення/виведення стане перевагою серверної частини. Підтримка node.js для авторизації, CORS, валідації, логування, зробить систему безпечнішою.

Серверна частина відповідатиме за:

— авторизацію користувачів, генерацію токенів;

— CRUD-операції з базою даних;

— перевірку прав доступу згідно з роллю;

— надання звітності та статистики.

2.6 Вибір хмарного сховища даних для розроблюваної системи

У сучасних системах важливу роль відіграє можливість надійного, захищеного та централізованого зберігання даних, доступ до якого є в будь-який момент. Особливо це актуально для онлайн систем, які працюють з великою

кількістю користувачів і документів, що потребують доступу з різних пристроїв і місць.

Для контролю централізованого зберігання даних використовують хмарні сервіси та технології.

Amazon Web Services (AWS), це платформа компанії Amazon, яка є лідером хмарного ринку. Має широкий набір сервісів, включаючи Amazon S3, яка відповідає за зберігання файлів, EC2, серверні структури, IAM, контроль доступу до системи. Amazon надає сучасні та гнучкі інструменти. Проте ефективне використання AWS потребує глибокого розуміння його структури, архітектури та принципів взаємодії між внутрішніми сервісами. Інтерфейс платформи є складним для новачків, а налаштування, такі як політики безпеки, правила маршрутизації чи керування обліковими записами, вимагають додаткових знань. Через це розробникам часто доводиться проходити офіційні сертифікаційні курси або вивчати велику кількість технічної документації. Окрім цього, більшість функцій AWS доступні на платній основі, що потребує постійного контролю витрат і знання тарифікації [17].

Microsoft Azure — хмарна платформа від Microsoft, яка інтегрується з Windows-середовищем і надає інструменти для створення вебзастосунків, обробки даних, хостингу, а також має набір інструментів OneDrive SDK [18].

OneDrive SDK дозволяє розробникам інтегрувати функціональність хмарного сховища Microsoft OneDrive у свої застосунки. За допомогою SDK можна реалізувати збереження, завантаження, перегляд, редагування та видалення файлів безпосередньо з хмарного сховища користувача [19].

Важливо зазначити, що OneDrive SDK підтримує як особисті акаунти, так і корпоративні акаунти Microsoft 365. Проте на практиці між цими двома типами облікових записів існують відмінності. Особисті акаунти надають обмежений обсяг сховища та функцій, але доступні всім користувачам. Корпоративні акаунти, які заадмініструються навчальним закладом, мають розширені політики безпеки, які можуть обмежувати деякі дії, більше місця, а також дозволяють централізований контроль і груповий доступ до документів.

Google Cloud Platform — це комплекс хмарних сервісів, розроблений компанією Google, який функціонує на тій самій інфраструктурі, що використовується для власних продуктів Google, зокрема Google Search і YouTube. Платформа пропонує не лише інструменти для керування ресурсами, а й модульні сервіси для обчислень, зберігання даних, аналізу, машинного навчання тощо. Для реєстрації у GCP необхідна банківська картка або рахунок [20].

Google Cloud Platform є складовою більш широкої екосистеми Google Cloud, до якої також входять Google Workspace, корпоративні рішення на базі Android і Chrome OS, а також API для роботи з машинним навчанням та сервісами Google Maps.

Для організації сховища даних для розроблюваної онлайн системи обрано Google Cloud Console. Завдяки використанню Google Drive API та налаштуванням через Google Cloud Console відповідальні особи можуть завантажувати документи безпосередньо в спеціальну директорію на Google Drive [21]. Деканат отримає доступ до перегляду файлів актів про порушення або заохочення і, таким чином, будем забезпечено централізоване зберігання, захист і можливість швидкого доступу до документів через систему.

Взаємодія з Google Drive здійснюється через сервер Node.js за допомогою OAuth2-авторизації і відповідних SDK для роботи з файлами.

2.7 Проектування структури бази даних для онлайн системи контролю проживання студентів у гуртожитках

У якості системи керування базами даних використовується безкоштовна система керування реляційними базами даних MySQL з відкритим вихідним кодом. Вона була створена як відкрита альтернатива комерційним СУБД і з часом стала однією з найпопулярніших у світі [22].

Вибір MySQL обумовлений її стабільною продуктивністю навіть при роботі з великими обсягами даних, наявністю підтримки зовнішніх ключів, індексів і транзакцій, а також широкою інтеграційною підтримкою в середовищі Node.js за допомогою бібліотек, зокрема mysql2.

Необхідно врахувати, що кожна роль користувача в системі має чітко визначений і обмежений набір дозволених дій, що забезпечує розмежування прав доступу та запобігає несанкціонованому перегляду або редагуванню даних іншими користувачами. Такий підхід відповідає принципам інформаційної безпеки та конфіденційності, особливо важливим при роботі з персональними даними студентів і службовою інформацією. У системі передбачено декілька рівнів ролей, такі як: представник від студентів, адміністратор системи та адміністратор бази даних. Кожна з цих ролей має доступ виключно до тих можливостей системи, які відповідають її повноваженням та обов’язкам. На рисунку 2.2 зображено запропоновану схему ролі користувачів в системі обліку та доступні їм функції.

Структура бази даних побудована відповідно до вимог системи і забезпечує логічне групування даних, зв'язність між таблицями та можливість подальшого масштабування. ER-діаграма бази даних наведена на рисунку Г.1.

Модель даних охоплює основні аспекти роботи системи, включаючи облік студентів, їх переміщення між кімнатами, реєстрацію порушень, створення актів, збереження рейтингів, архівування студентів та інше. Усі таблиці мають чіткі зв’язки, що дає змогу ефективно реалізовувати складні аналітичні запити та забезпечити узгодженість даних.

Створено наступну логіку зв’язків:

— студент (students) проживає в певній кімнаті (rooms), яка належить гуртожитку (dormitories);

— кожне порушення (violations) прив’язується до конкретного студента;

— запропоновані акти (proposed_acts) формуються для студента на основі зафіксованих порушень;

—акти (acts) створюються на основі порушень і затверджуються адміністратором;

— рейтинги (rating) пов’язані зі студентами та оновлюються на основі історії їх поведінки (порушень або заохочень);

— переміщення (relocations) фіксують зміну кімнат студента та зберігають історію переселень.

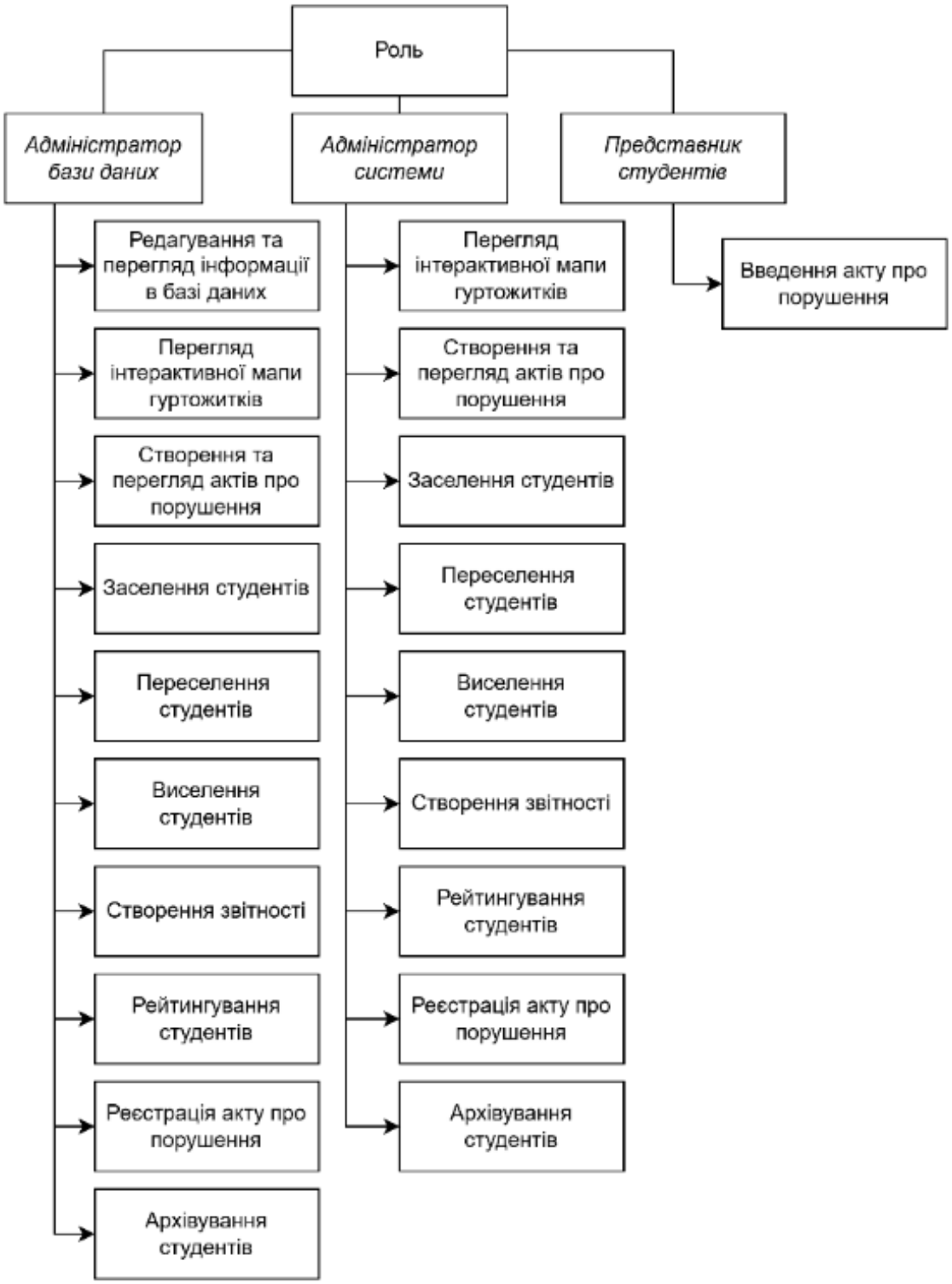

Рисунок 2.2 — Ролі користувачів в системі обліку та доступні їм функції

Отже, результати створюють міцну технічну основу для безпосередньої реалізації, тестування та впровадження комп’ютерної онлайн системи у практичну діяльність навчального закладу.

2.8 Проектування архітектурного рішення онлайн систем контролю проживання студентів у гуртожитках

Оскільки, система побудована за клієнт-серверною архітектурою та має чітке розділення на три основні рівні: клієнтський інтерфейс, серверну частину та базу даних. Взаємодія між ними організована через API, а передача даних здійснюється у форматі JSON. Схема взаємодії користувача з компонентами системи наведено на рисунку 2.3.

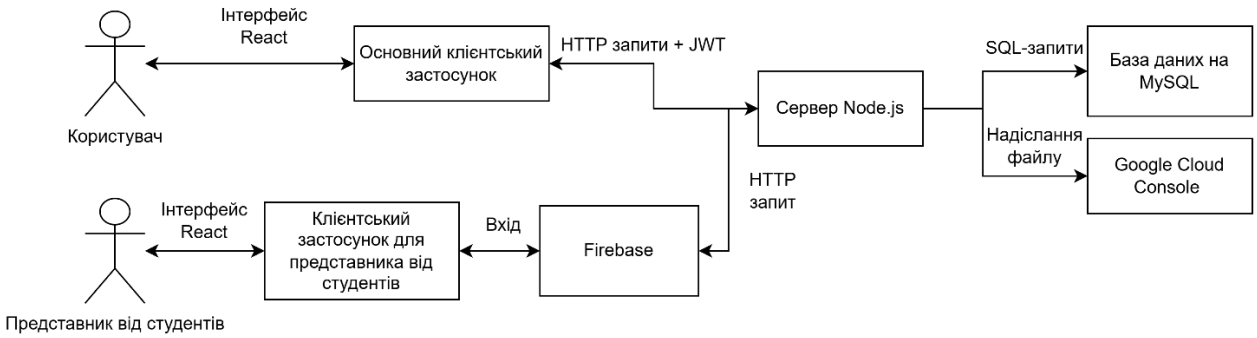

Рисунок 2.3 — Схема взаємодії користувача з компонентами системи

Система складається з трьох основних логічних рівнів:

— клієнтський рівень (Frontend);

— серверний рівень (Backend / API);

— рівень зберігання даних (Database).

Архітектурні рівні системи зображено на рисунку Г.2.

На клієнтському рівні реалізовано інтерактивний інтерфейс користувача, який працює у браузері та забезпечує зручну взаємодію з системою. Інтерфейс забезпечує взаємодію з користувачем та відправляє запити до сервера через REST API. Усі компоненти клієнтської частини побудовані з використанням бібліотеки React, що забезпечує гнучкість, повторне використання елементів інтерфейсу, легкість масштабування та підтримку сучасних практик розробки.

Клієнт надсилає HTTP-запити до API-сервера (GET, POST, PUT, DELETE) і отримує відповіді у форматі JSON. Візуальна частина адаптована для настільних і мобільних пристроїв.

Сервер відповідає за бізнес-логіку системи, перевірку прав доступу, обробку запитів, взаємодію з базою даних і формування відповідей.

До основних функцій серверного рівня належать: авторизація та автентифікація користувачів; управління студентами, кімнатами, гуртожитками, а саме CRUD-операції; обробка актів порушень і рейтингів; формування зведеної інформації та звітів; обробка переселень і архівація студентів.

Використовується підхід коли кожен ресурс, наприклад, студент, гуртожиток, акт, має набір маршрутів, наприклад /students, /rooms, /violations, які виконують відповідні дії.

Крім обробки запитів і взаємодії з MySQL, серверна частина включає підтримку зовнішніх сервісів, зокрема Google Cloud Console та Firebase Authentication, які використовуються для авторизації представників студентського самоврядування.

Авторизація представників студентства, наприклад, членів студради, здійснюється за допомогою Firebase Authentication, який інтегрується із клієнтською частиною та Google акаунтами. Студенти проходять автентифікацію через Google (OAuth 2.0), після чого отримують безпечний токен доступу. Цей токен надсилається до серверу, де проходить валідацію через Firebase SDK [23].

Система надає представникам студентства обмежений рівень доступу, наприклад, лише пропонування порушення чи заохочення для конкретного студента.

Google Cloud Console використовується для налаштування авторизаційних облікових записів, OAuth-клієнтів, ключів API та дозволів доступу до сервісів Google Drive. Через цю платформу здійснюється керування сервісами, які дозволяють серверу з Node.js взаємодіяти з Google Drive, завантажувати файли та прив’язувати їх до записів у базі даних.

Зберігання структурованої інформації реалізується у реляційній базі даних MySQL. Вона містить таблиці, пов’язані зовнішніми ключами.

Сервер взаємодіє з базою через SQL-запити, забезпечуючи збереження, пошук, фільтрацію, оновлення і видалення записів.

Взаємодія в системі побудована за моделлю «клієнт-сервер-база даних».

Користувач взаємодіє з інтерфейсом у браузері. Інтерфейс відправляє запити до сервера. Сервер обробляє запит, звертається до бази даних, виконує операцію і формує відповідь. Відповідь надсилається назад у клієнтську частину, де дані оновлюються на екрані. Усі запити захищені механізмом авторизації, а доступ до функцій системи контролюється відповідно до ролі користувача.

Перевагами обраної архітектури є модульність, масштабованість та безпека. Кожен рівень може оновлюватися незалежно. Також легко додати нові функції, гуртожитки, або мобільну версію. В плані безпеки сервер повністю контролює доступ до даних.

2.9 Вимоги до апаратного забезпечення онлайн системи контролю проживання студентів у гуртожитках

Апаратне забезпечення — це сукупність апаратних засобів, необхідних для функціонування інформаційної системи. Оскільки обрано клієнт-серверної трьохрівневу архітектуру системи, яка передбачає наявність окремих компонентів для серверної частини, клієнтської частини та інфраструктурного середовища.

Зважаючи на сучасні технології, система не має надмірних апаратних вимог, але для коректної та стабільної роботи слід дотримуватись мінімальних і рекомендованих характеристик для кожної складової системи.

Серверна частина обробляє запити, взаємодіє з базою даних та забезпечує API-доступ клієнтів до системи. Апаратні вимоги до обладнання для серверної частини наведено на таблиці 2.1.

Клієнтська частина запускається у браузері та відображає інтерфейс системи та взаємодіє з API. Апаратні вимоги до комп’ютера для клієнтської частини наведено на таблиці 2.2.

Для зберігання документів система використовує інтеграція з Google Cloud Console. Використання цієї платформи знімає потребу у власному файловому сервері, проте потребує стабільного інтернет-з’єднання та наявності Google-акаунту з відповідними налаштуваннями.

Таблиця 2.1 — Апаратні вимоги до обладнання для серверної частини

Параметр	**Мінімальні вимоги**	**Рекомендовані вимоги**
Процесор	2 ядра (1.8–2.0 GHz)	4 ядра (2.4 GHz і вище)
Оперативна пам’ять	4 ГБ	8 ГБ
Жорсткий диск	20 ГБ вільного місця	SSD 50 ГБ і більше
ОС	Linux Ubuntu 20.04 / Windows 10+	Linux Ubuntu Server 22.04 LTS

Таблиця 2.2 — Апаратні вимоги до комп’ютера для клієнтської частини

Параметр	**Мінімальні вимоги**	**Рекомендовані вимоги**
Процесор	2 ядра (1.6 GHz)	4 ядра (2.0 GHz і вище)
Оперативна пам’ять	4 ГБ	8 ГБ
Відеокарта	Інтегрована	Інтегрована / базова дискретна
Дисплей	1366x768	Від Full HD (1920x1080)
ОС	Windows 10 / macOS / Linux	Windows 11 / macOS Ventura / Ubuntu 22.04
Браузер	Chrome 90+, Firefox 88+, Edge 90+	Chrome останньої версії
Швидкість інтернету	Мін. 5 Мбіт/с	25 Мбіт/с і більше

Отже, наведені апаратні характеристики повністю відповідають мінімальним та рекомендованим вимогам для стабільної роботи клієнтської та серверної частин системи.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ОНЛАЙН СИСТЕМИ КОНТРОЛЮ ПРОЖИВАННЯ СТУДЕНТІВ У ГУРТОЖИТКАХ

3.1 Інструментальні засоби та середовища розробки

Для реалізації системи обрано оптимальні інструментальні засоби та середовища розробки, які забезпечують ефективну організацію процесу програмування, тестування та підтримки системи.

Visual Studio Code — це основне середовище для розробки проекту. Являє собою редактор вихідного коду, розроблений компанією Microsoft на базі Electron, доступний для Windows, Linux і macOS. Він забезпечує широкі можливості, включаючи налагодження, підсвічування синтаксису, інтелектуальне автозаповнення, фрагменти коду, рефакторинг та інтеграцію з Git [24].

Редактор підтримує гнучке налаштування: користувачі можуть змінювати теми, комбінації клавіш, параметри середовища, а також встановлювати розширення для розширення можливостей системи.

Postman — це програма для роботи з API, яка дозволяє розробникам швидко створювати, тестувати, надсилати та документувати HTTP-запити. Вона підтримує різні методи, такі як GET, POST, PUT та інші, збереження колекцій запитів, авторизацію, змінні середовищ і спільну роботу в команді. Postman особливо корисний під час створення або налагодження серверної частини застосунків [25].

GitHub — це один із найпопулярніших вебсервісів для спільної розробки програмного забезпечення, що працює на основі системи контролю версій Git. Платформа була створена компанією GitHub, Inc [26]. GitHub підтримує як безкоштовні, так і платні плани. Усі основні функції, зокрема підтримку SSL, доступні безкоштовно для проєктів із відкритим кодом. Використовується для контролю версіями проекту.

Для створення, моделювання та візуалізації структури бази даних використовується MySQL Workbench — безкоштовний інструмент для візуального проектування баз даних, який поєднує в собі можливості моделювання, створення, адміністрування та супроводу баз даних у єдиному середовищі для СУБД MySQL [27].

MySQL Workbench доступний у двох версіях: безкоштовна Community Edition, яка розповсюджується за ліцензією GNU GPL, та комерційна Standard Edition, що надається за передплатою і містить розширені інструменти для підвищення ефективності роботи розробників і адміністраторів баз даних.

3.2 Реалізація бази даних онлайн системи контролю проживання студентів у гуртожитках

Для забезпечення централізованого зберігання та обробки інформації про студентів, гуртожитки, кімнати, акти порушень і рейтинги у системі використовується реляційна база даних, реалізована за допомогою MySQL. Проєктування бази даних здійснено у середовищі MySQL Workbench.

Таблиця users призначена для зберігання даних про зареєстрованих користувачів системи. Структуру полів таблиці user наведено у таблиці 3.1.

Таблиця 3.1 — Структура полів таблиці user

Поле	**Тип**	**Опис**
id	int	Первинний ключ, ідентифікатор користувача
username	varchar(50)	Унікальне ім’я користувача
password	varchar(255)	Хеш пароля
role	varchar(50)	Роль користувача в системі, admin, dean, responsible_person
created_at	timestamp	Дата створення запису
updated_at	timestamp	Дата останнього оновлення

Таблиця students містить інформацію про студентів, що проживають в гуртожитках. Структуру полів таблиці students наведено у таблиці 3.2.

Таблиця archived_students містить інформацію про студентів, які вже виселені або переведені до архіву. Структуру полів таблиці archived_students наведено у таблиці 3.3.

Таблиця 3.2 — Структура полів таблиці students

Поле	**Тип**	**Опис**
id	int	Первинний ключ
pib	varchar(255)	ПІБ студента
group	varchar(50)	Код та номер академічної групи
room_id	int	Прив’язка до кімнати
rating	int	Рейтинг студента
benefit	varchar(255)	Наявність пільг
violations	int	Кількість порушень
acts_count	int	Кількість актів
additional_info	text	Додаткова інформація про студента
needs_relocation	tinyint(1)	Чи потребує студент переселення

Таблиця 3.3 — Структура полів таблиці archived_students

Поле	**Тип**	**Опис**
id	int	Первинний ключ
pib	varchar(255)	ПІБ студента
group	varchar(50)	Код та номер академічної групи
rating	int	Рейтинг студента на момент архівації
benefit	varchar(255)	Наявність пільг
acts_count	int	Кількість зареєстрованих актів
violations	int	Кількість зафіксованих порушень
room_id	int	Кімната, в якій проживав студент
archived_at	datetime	Дата архівації запису
can_be_rehoused	tinyint(1)	Можливость повторного поселення

Таблиця dormitories зберігає інформацію про гуртожитки. Структуру полів таблиці dormitories наведено у таблиці 3.4.

Таблиця 3.4 — Структура полів таблиці dormitories

Поле	**Тип**	**Опис**
id	int	Первинний ключ
name	varchar(255)	Назва гуртожитку
address	varchar(255)	Адреса гуртожитку
floors	int	Кількість поверхів

Таблиця rooms описує кімнати в гуртожитках та їх стан. Структуру полів таблиці rooms наведено у таблиці 3.5.

Таблиця 3.5 — Структура полів таблиці rooms

Поле	**Тип**	**Опис**
id	int	Первинний ключ
dormitory_id	int	Зовнішній ключ на гуртожиток
floor_number	int	Номер поверху
room_number	varchar(10)	Номер кімнати
beds_count	int	Кількість ліжко-місць у кімнаті
beds_description	varchar(255)	Додатковий опис ліжок (наприклад, зайнятість)

Таблиця violations зберігає інформацію про порушення або заохочення студентів. Структуру полів таблиці violations наведено у таблиці 3.6.

Таблиця 3.6 — Структура полів таблиці violations

Поле	**Тип**	**Опис**
id	int	Первинний ключ
student_id	int	Зовнішній ключ на students.id
date	date	Дата порушення
reason	text	Опис причини
file_url	varchar(255)	Посилання на документ (фото/акт)
type	enum('act','warning')	Тип запису: порушення (акт) або попередження

Таблиця ratings містить автоматично сформовані рейтинги студентів Структуру полів таблиці ratings наведено у таблиці 3.7.

Таблиця 3.7 — Структура полів таблиці ratings

Поле	Тип	Опис
id	int	Первинний ключ
student_id	int	Зовнішній ключ на students.id
points	int	Кількість нарахованих балів
description	text	Опис документу
file_url	varchar(255)	Посилання на документ
created_at	timestamp	Дата та час запису

Таблиця relocations фіксує переселення студентів з однієї кімнати в іншу. Структуру полів таблиці relocations наведено у таблиці 3.8.

Таблиця 3.8 — Структура полів таблиці relocations

Поле	Тип	Опис
id	int	Первинний ключ
student_id	int	Зовнішній ключ на students.id
previous_room_id	int	Зовнішній ключ на попередню кімнату
new_room_id	text	Зовнішній ключ на нову кімнату
created_at	timestamp	Дата переселення

Таблиця proposed_acts зберігає акти про порушення або заслуги, запропоновані відповідальною особою і очікують рішення декана. Структуру полів таблиці proposed_acts наведено у таблиці 3.9.

Таблиця 3.9 — Структура полів таблиці proposed_acts

Поле	Тип	Опис
id	int	Первинний ключ
student_id	int	Зовнішній ключ на students.id
date	date	Дата створення документу
reason	text	Причина порушення або заслуги
file_url	varchar(255)	Посилання на прикріплений документ у Google Drive
status	enum('pending', 'approved', 'rejected')	Поточний статус акту
created_at	timestamp	Дата переселення

Отже, розроблена та реалізована модель бази даних забезпечує логічну організацію зберігання інформації, мінімізує дублювання даних, дозволяє ефективно проводити запити та формувати аналітичні звіти для адміністрації гуртожитків і деканату.

3.3 Реалізація клієнтської частини онлайн систем контролю проживання студентів у гуртожитках

Клієнтська частина реалізована у вигляді односторінкового вебзастосунку (SPA) з використанням React.js [28].

SPA (Single Page Application) — це вебзастосунок або вебсайт, який динамічно завантажує контент і оновлює інтерфейс без повного перезавантаження сторінки. Користувач працює на одній HTML-сторінці, де змінюється лише вміст за допомогою JavaScript або додаткових бібліотек. На відміну від традиційних багатосторінкових застосунків, у SPA вся логіка навігації, завантаження даних і відображення виконується на стороні клієнта.

У системі обліку проживання студентів важливо забезпечити швидку реакцію на дії користувача, інтерактивну мапу без перезавантаження сторінки, динамічні таблиці з фільтрами, сортуванням та діалогами, роботу з JWT-сесіями без постійного перезавантаження та гнучке розмежування доступу для декана, адміністрації, представників студентів. Single Page Application з React дозволяє реалізувати все це максимально ефективно.

Застосунок розроблено з урахуванням потреб різних ролей користувачів та забезпечує доступ до можливостей системи відповідно до прав доступу.

Архітектура клієнта побудована за принципами: Component-based, Routing with React Router, State management.

По-перше, Component-based — інтерфейс розбитий на самостійні компоненти, які відповідають за окремі частини інтерфейсу: таблиці, панелі, форми, карти тощо.

По-друге, Routing with React Router — маршрутизація організована за ролями користувачів: кожен маршрут захищено логікою перевірки токена та ролі.

По-третє, State management — загальний стан додатку зберігається через useContext з можливістю подальшої міграції на Redux або Zustand.

Розглянемо ключові інтерфейсні модулі та взаємодії між ними. Схема структури клієнта наведено на рисунку Г.3.

Модуль авторизації та входу для основної сторінки системи вхід відбувається через форму з логіном та паролем. Графічний інтерфейс сторінки входу системи наведено на рисунку Г.4. Після успішного входу користувач отримує доступ до системи, а саме до головної сторінки.

Для представників від студенства, створено окремий програмний модуль реєстрації порушень правил проживання студентів у гуртожитку. Вхід відбувається через введення електронної пошти та паролю чи за допомогою акаунта Google. Механізм автентифікації реалізовано з використанням Google Firebase. Графічний інтерфейс входу в програмний модуль реєстрації порушень правил проживання студентів у гуртожитку наведено на рисунку Г.5.

Після успішної авторизації представника від студентів отримує доступ до форми подання порушення чи заслуги студента. Графічний інтерфейс форми подання програмного модуля реєстрації порушень правил проживання студентів у гуртожитку наведено на рисунку Г.6.

Мапа гуртожитку є одним із ключових візуальних інструментів та особливістю системи. Вона реалізована у вигляді SVG, який відображає поверхи та кімнати будівлі з можливістю перемикання між поверхами. Графічний інтерфейс інтерактивної мапи гуртожитку наведено на рисунку Г.7. Лістінг програмного коду файлу HomePage.js наведено в додатку Д.

Кожна кімната відображається у вигляді мітки, колір якого залежить від її статусу. Якщо кімната немає порушників, — вона зеленого кольору. Якщо в кімнаті мало кількість порушень, тоді вона жовта. Якщо кількість порушень перевищує 3, тоді буде офарбована в червоний колір. Також, є сірий стан кімнати, який означає, що кімната неактивна (пуста) або недоступна для студентів.

Візуалізація в реальному часі дозволяє помічати кожну зміну в базі та миттєво відображається на мапі. Інтерактивність мапи полягає у тому що, при натисканні на кімнату відкривається вікно з інформацією про кількість ліжок, зайнятість і список студентів. Модальне вікно кімнати наведено на рисунку 3.1.

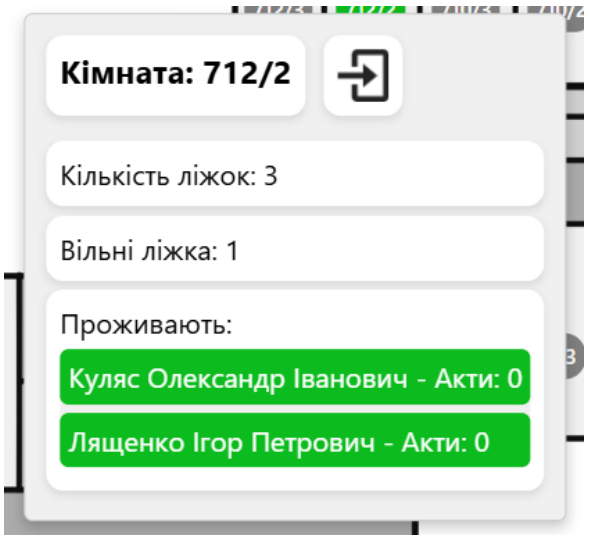

Рисунок 3.1 — Модальне вікно кімнати

Перехід між поверхами відбувається перемиканням між поверхами гуртожитку, що дозволяє швидко знайти потрібну кімнату.

Також мапа являє собою вхідною точкою для формування звітів про зайнятість кімнат, наповненість гуртожитку, потребу в переселенні тощо.

Сторінка «Інші гуртожитки» надає доступ до таблиці з фільтрацією за групами, рейтингами, гуртожитками. Кожен рядок веде до детального профілю студента. Графічний інтерфейс сторінки «Інші гуртожитки» зображено на рисунку Г.8.

Профіль студента відображає інформацію про поселення, історію переселень, рейтинги, акти порушень та документи. Також передбачена можливість редагування даних. Графічний інтерфейс профілю студента наведено на рисунку Г.9.

Панель актів відображає список актів, які вже наявні в системі. Також адміністратор може додати новий акт, запропонувати акт та розглянути запропоновані вже студентами акти.

Система включає три основні типи форм для керуванням поселенням студентів.

Форма поселення відповідає за поселення студента в гуртожиток та заповнення базової інформації дані про нього, такі як ПІБ, група, пільги, додаткова

інформація. Після введення інформації адміністратор має вибрати кімнату в гуртожитку та зберегти дані, після чого дані відправляться на сервер.

Далі, форма переселення відповідає за переселення студентів в іншу кімнату чи загальному переселенню. Відображає поточну кімнату, надає вибір нової доступної кімнати або залишити студента в цій. Графічний інтерфейс форми поселення наведено на рисунку Г.10.

Форма виселення відповідає за виселення студента з гуртожитку.

Після виселення студент потрапляє в архів, історія зберігається. З архіву наявна можливість повернення студента в гуртожиток, проте є можливість обмежити це право, під час виселення.

Система підтримує автоматичне формування звітів для адміністрації. Для звітів наявна можливість фільтрувати за піб, датою, описом чи файлом та експортувати у Excel файл для подальшого аналізу чи подання керівництву.

Звітність формує звіт щодо порушень — кількість і тип порушень по кожному студенту, за період.

Звітність (рейтинг) формує звіт рейтингу по групах — середній бал за рейтингом, кількість заохочень/порушень.

Сторінка «Рейтинг» дозволяє додавати рейтинг студента, експортувати весь рейтинг в Excel файл та надає можливість перегляду та фільтрування рейтингу. Графічний інтерфейс сторінки «Рейтинг» наведено на рисунку Г.11.

Сторінки «Керування актами», «Керуванням кімнатами» та «Керування гуртожитками» доступна лише для користувачів з роллю адміністратор бази даних, тому що ці модулі напряму взаємодіють з даними з бази даних.

Сторінка «Керування гуртожитками» надає доступ до створення, редагування та видалення гуртожитків. Графічний інтерфейс сторінки «Керування гуртожитками» наведено на рисунку Г.12.

Сторінка «Керування кімнатами» надає доступ до створення, редагування ,видалення та зазначення кількості ліжок з їх описом для кімнат гуртожитку. Графічний інтерфейс сторінки «Керування кімнатами» наведено на рисунку Г.13.

Сторінка «Керування актами» надає доступ до списку усіх актів у статусі, фільтрація за типом, студентом, датою подання, та перегляд супровідного документа. Також наявна функція редагування та видалення вже наявного акту. Графічний інтерфейс сторінки «Керування актами» наведено на рисунку Г.14.

Інтерфейс форм дозволяє прикріпити файл до порушення або заслуги. Завантаження реалізовано через форму з переглядом, після чого файл надсилається на сервер та зберігається на Google Drive. Користувач отримує посилання на документ, який можна переглянути в новому вікні.

Завдяки гнучкій верстці, застосунок правильно відображається на смартфонах та планшетах.

Отже, клієнтська частина програмного забезпечення реалізована у вигляді односторінкового вебзастосунку з використанням бібліотеки React.js, що забезпечує швидкодію, інтерактивність та зручність взаємодії для кінцевого користувача. Завдяки компонентній архітектурі та клієнтській маршрутизації, система легко масштабована, підтримує розмежування прав доступу й адаптивне відображення на різних пристроях.

Отже, в результаті реалізації системи досягнуто високої інтерактивності інтерфейсу, гнучкого управління даними та повної підтримки життєвого циклу поселення студентів — від реєстрації до виселення з архівацією даних. Реалізація всіх функціональних блоків підтверджує готовність системи до практичного використання у вищому навчальному закладі.

3.4 Розробка серверної частини та API онлайн систем контролю проживання студентів у гуртожитках

Серверна частина системи розроблена на основі Node.js. Для організації маршрутизації, обробки HTTP-запитів і створення RESTful API було використано Express.js, що забезпечило модульну структуру та спрощену інтеграцію з базою даних, системами авторизації та сторонніми сервісами.

Серверну частину побудовано за принципами MVC (Model-View-Controller).

Архітектурний патерн Model-View-Controller (MVC) розділяє додаток на три основні групи компонентів: Моделі, Представлення та Контролери. Цей патерн допомагає досягти поділу завдань. Запити користувача спрямовуються до контролера, який відповідає за роботу з моделлю для виконання дій користувача та/або отримання результатів запитів. Контролер обирає подання, яке буде показано користувачеві, і надає йому всі необхідні дані Моделі. Реалізація патерну MVC наведено на рисунку 3.2 [29].

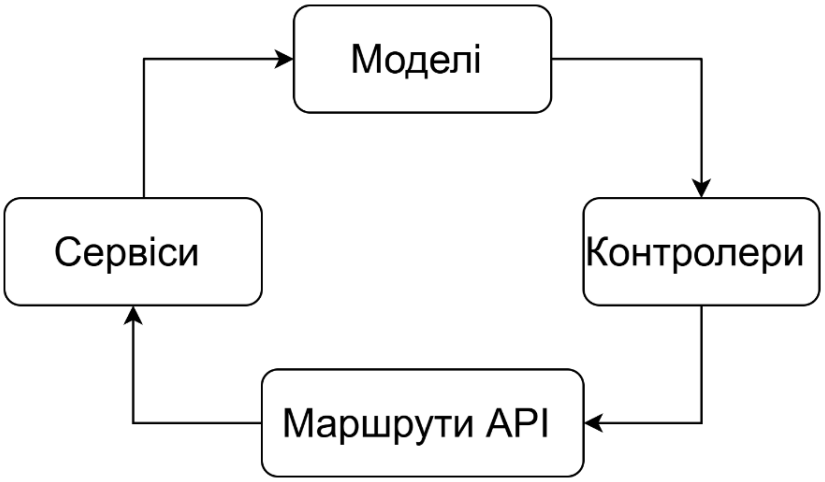

Рисунок 3.2 — Реалізація патерну MVC

Моделі відображають структуру таблиць бази даних, наприклад, Student, Room, Dormitory, Act.

Контролери містять логіку обробки запитів та відповіді на дії клієнта.

Маршрути описують кінцеві точки API.

Сервіси обробляють логіку бізнес-рівня, наприклад, завантаження документів на Google Drive чи обчислення рейтингу.

Структура серверної частини включає такі компоненти: Маршрути, Middleware, головний вхідний файл, файл конфігурації для бази даних, файл конфігурації середовища .env, інтеграція з Google Drive та тимчасова папка для завантаження.

Маршрути (Routes) — це окремі файли, що відповідають за обробку HTTP-запитів до конкретних ресурсів, наприклад, /students, /violations, /dormitories. Вони

визначають кінцеві точки REST API та передають запити до відповідних контролерів.

Middleware — це функції проміжної обробки, які виконуються між запитом клієнта та відповіддю сервера. До них належать перевірка автентифікації, обробка CORS-запитів, валідація даних тощо.

Головний вхідний файл (index.js) — це точка входу в серверну частину, де ініціалізується Express-додаток, підключаються маршрути, middleware, конфігураційні змінні, обробники помилок і запуск сервера.

Файл конфігурації для бази даних — це окремий модуль, що встановлює з'єднання з базою даних із використанням драйвера або іншого засобу.

Файл конфігурації середовища .env — це файл, який зберігає змінні середовища, такі як порт, URI бази даних, секретні ключі та токени. Це забезпечує безпечне та зручне конфігурування системи під різні середовища.

Інтеграція з Google Drive — це окремий модуль, який реалізує логіку завантаження актів, документів та інших файлів на хмарне сховище Google Drive через API, з подальшим збереженням посилань у базі даних.

Тимчасова папка для завантаження використовується для тимчасового збереження файлів, що надходять від користувача, перед їх передачею в хмару або базу.

Запропоновану структуру серверної частини наведено на рисунку Г.15.
База даних являє собою централізоване сховище для всіх сутностей системи. Сервер взаємодіє з нею через окремий рівень абстракції.

Клієнтська частина надсилає запити до серверного API, отримує оброблені відповіді та відображає їх користувачеві у зручному інтерфейсі.

Сервер підтримує RESTful API. В загальному REST (Representational State Transfer) — це архітектурний стиль для побудови мережевих протоколів, що забезпечують доступ до інформаційних ресурсів [30].

Передача даних у REST здійснюється через обмежену кількість стандартних форматів, таких як HTML, XML або JSON. REST-протоколи, включно з HTTP, мають відповідати кільком важливим вимогам: підтримувати хешування, бути

незалежними від транспортного рівня мережі, а також бути безстанними — тобто не зберігати інформацію між запитами та відповідями. Такий підхід сприяє масштабованості систем і дає змогу гнучко адаптувати їх до нових потреб.

Ендпоінти — це кінцеві точки взаємодії з API, через які інші додатки, вебсайти чи програмні сервіси отримують доступ до даних або сервісів, що надає система [31].

Ендпоінти зазвичай представлені у вигляді URL-адрес, які визначають шлях до конкретного ресурсу, а також тип відповіді. Через них можна, наприклад, отримати інформацію, відправити запит на створення або зміну даних, або ініціювати певні дії всередині системи. Через критичну роль у доступі до системи, важливо забезпечити належний рівень захисту ендпоінтів, щоб запобігти несанкціонованому доступу до важливої інформації.

Розглянемо, які саме ендпоінти реалізовано в системі та які функції вони виконують.

Ендпоінти для таблиці студентів:

— GET /students — отримання списку всіх студентів із фільтрацією;

— POST /students — створення нового запису про студента;

— PUT /students/:id — редагування даних про студента;

— DELETE /students/:id — архівація (логічне видалення) студента.

Ендпоінти для таблиць гуртожитків та кімнат:

— GET /dormitories — список усіх гуртожитків;

— GET /rooms?dormitoryId=... — кімнати певного гуртожитку;

— POST /rooms — додавання нової кімнати;

— PUT /rooms/:id — редагування даних про кімнату.

Ендпоінти для таблиць актів та порушень:

— POST /acts — створення акта порушення або заохочення;

— GET /acts/pending — отримання актів, що очікують на рішення;

— PATCH /acts/:id/status — оновлення статусу акта деканом (approved, rejected).

Ендпоінти для таблиці рейтингу:

— GET /rating/:studentId — отримання рейтингу студента;

— POST /rating/recalculate — перерахунок рейтингу вручну або за подією.

Ендпоінти для таблиці авторизації користувача:

— POST /login — автентифікація користувача за логіном і паролем.

Також захищені маршрути перевіряють JWT-токен у заголовку Authorization.

JWT (JSON Web Token) — це відкритий стандарт (RFC 7519), що описує компактний і зручний формат для безпечного обміну інформацією між сторонами у вигляді JSON-об'єкта. У системі він слугує засобом автентифікації користувачів і контролю доступу до захищених частин API [32].

Процес автентифікації відбувається коли користувач вводить логін і пароль на клієнтській частині та надсилає POST /login запит на сервер. Далі сервер перевіряє дані користувача у таблиці users. Якщо облікові дані коректні — генерується JWT-токен.

Після успішного входу в систему токен зберігається у локальному сховищі браузера та додається до кожного запиту до API у заголовку.

Задля безпеки термін дії токена має обмежений час дії (близько 5 годин), що дозволяє зменшити ризик викрадення витику даних. При виході з системи токен просто видаляється на клієнті. Секретний ключ зберігається тільки на сервері, у файлах .env.

Оскільки використовується MySQL як основна система керування базами даних, взаємодія реалізована через бібліотеку mysql2 з підтримкою запитів.

Модулі доступу до даних дозволяють уникати SQL-ін’єкцій через запити з параметрами, використовувати транзакції при складних операціях.

Інтеграція з Google Cloud Console дає змогу завантажувати документи актів, довідок та порушень безпосередньо на Google Drive. Користувач надсилає потрібний файл через інтрефейс системи. Сервер за допомогою Google API створює відповідний файл на Google Drive. Після створення система зберігає отримане посилання на файл у базу даних, щоб надати доступ до документа. Перегляд файлу можливий через вебінтерфейс без повторного входу в Google [33].

Для отримання доступу до Google Cloud Console необхідно завантажити конфігураційний JSON-файл та підключити його до основного файлу сервера. Завантаження файлів на Google Drive обмежене відповідно до ролі користувача.

Отже, серверна частина задовольняє всі вимоги системи по безпеці та доступу до даних.

3.5 Захист даних та використання HTTPS в онлайн системі контролю проживання студентів у гуртожитках

У онлайн системі контролю проживання студентів у гуртожитках, зберігаються чутливі персональні дані користувачів: ПІБ, контактна інформація, історія поселення, порушення, рейтинг тощо. Тому забезпечення інформаційної безпеки є пріоритетом при реалізації серверної та клієнтської частини.

Для цього в системі передбачено використання протоколу HTTPS.

HTTPS (HyperText Transfer Protocol Secure) — це захищений варіант протоколу HTTP, який використовується для безпечної передачі даних через Інтернет. Синтаксично він подібний до звичайного HTTP, проте використовує інший порт за замовчуванням (443) і включає додатковий рівень шифрування та автентифікації, що працює поверх протоколу TCP [34].

Протокол HTTPS був розроблений компанією Netscape Communications Corporation з метою забезпечення конфіденційності та цілісності даних під час обміну інформацією. Сьогодні він є стандартом у сферах, де безпека має вирішальне значення — наприклад, при обробці платіжних даних чи захищеному вході в корпоративні системи.

Обмеження доступу до ендпоінтів реалізовано через middleware, яке перевіряє роль користувача, admin, dean, student.

Middleware — це функція, яка виконується під час обробки HTTP-запиту до того, як запит досягне ендпоінта. У Express.js middleware має доступ до об'єктів req, res і функції next(), яка викликає наступну обробку. Middleware використовується для автентифікації — перевірки, чи користувач має дійсний JWT-токен [35].

Для звернення до серверної частини використовується Axios. Axios — це популярна JavaScript-бібліотека для виконання HTTP-запитів, яка працює як у браузері, так і в Node.js. Вона є обгорткою над XMLHttpRequest і підтримує проміси (Promises), що дозволяє просто обробляти асинхронні операції [36].

Axios має низку переваг, серед яких — зручний синтаксис для виконання GET, POST, PUT і DELETE запитів, автоматична обробка JSON, а також можливість встановлення глобальних заголовків, таких як JWT-токен. Крім того, Axios підтримує інтерцептори, що дозволяють перехоплювати запити або відповіді, що особливо корисно для реалізації логіки авторизації. Помилки можна зручно обробляти за допомогою методу .catch(...), а також підтримуються функції тайм-аутів, скасування запитів і завантаження файлів.

Для забезпечення коректної роботи REST API системи було застосовано інструмент Postman. Тестування включало перевірку етапів як: авторизація користувача, робота з мешканцями, поселення і виселення, додавання порушень, формування рейтингу та формування звітів.

Розглянемо приклад запиту в Postman для авторизації та отримання токена, а також приклад захищеного запиту з використанням токена.

Авторизація користувача відбувається за допомогою методу POST за адресою http://localhost:5000/api/auth/login, в тіло запиту має бути вказано дані для входу, у відповідь ми отримуємо токен token, який потрібно використовувати для захищених маршрутів. На рисунку 3.3 наведено запит в Body. Також, на рисунку 3.4. відповідь запиту, а саме отриманий токен.

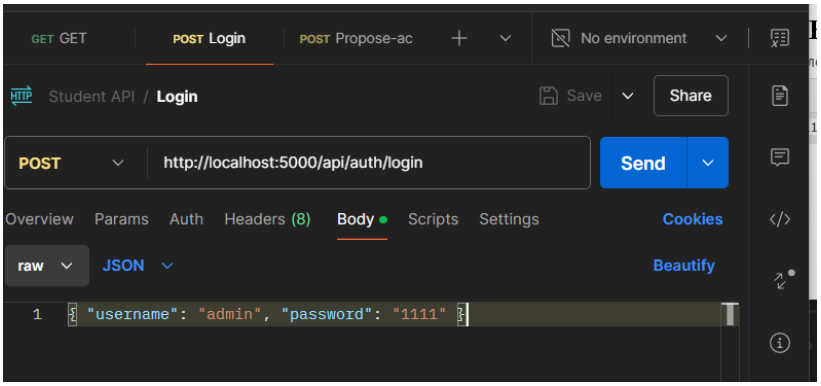

Рисунок 3.3 — Запит в Body для входу в Postman

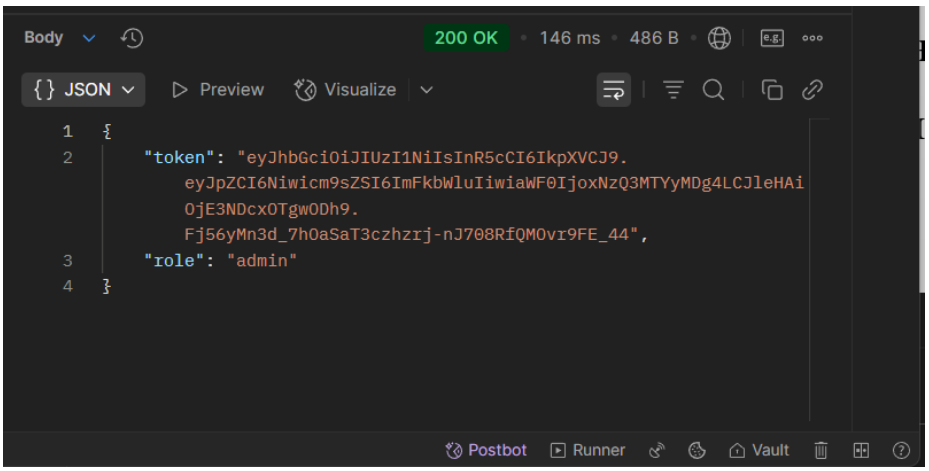

Рисунок 3.4 — Отриманий токен в Postman

Використаємо отриманий токен для того щоб запропонувати акт адміністрації. Для цього в Heder запиту необхідно вказати наш JWT токен та в Body вказати запит. Поле Heder з заповненим токеном наведено на рисунку 3.5, а вікно з запитом наведено на рисунку 3.6.

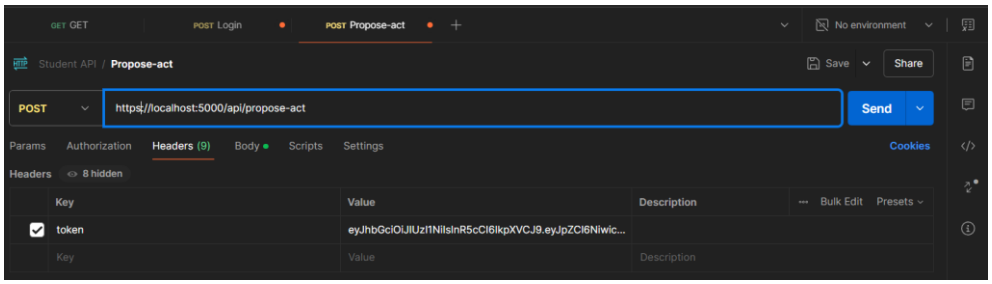

Рисунок 3.5 — Поле Heder з заповненим токеном в Postman

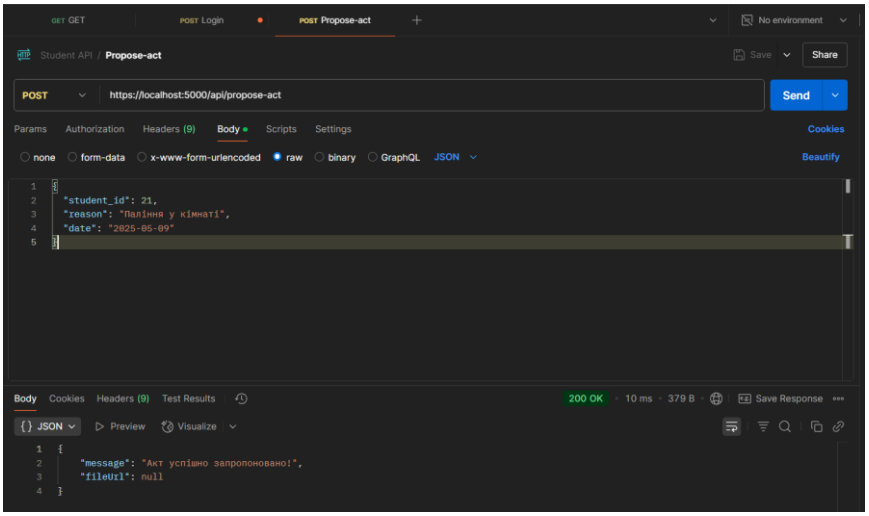

Рисунок 3.6 — Вікно з запитом для пропозиції акту в Postman

Отже, реалізація підтримки протоколу HTTPS дозволяє забезпечити надійне шифрування трафіку, запобігти перехопленню конфіденційної інформації та гарантувати цілісність переданих даних. Такий рівень безпеки є особливо важливим з огляду на збереження персональних облікових записів студентів, їх історії проживання, актів порушень та рейтингів. Застосування інструменту Postman для тестування API дозволило перевірити стабільність серверної частини.

4 ТЕСТУВАННЯ ТА ВПРОВАДЖЕННЯ ОНЛАЙН СИСТЕМ КОНТРОЛЮ ПРОЖИВАННЯ СТУДЕНТІВ У ГУРТОЖИТКАХ

4.1 Тестування можливостей клієнтської частини онлайн систем контролю проживання студентів у гуртожитках

Для забезпечення стабільної та коректної роботи системи проводять тестування, яке має охоплювати ключові аспекти функціонування та взаємодії з користувачем. Існує декілька основних типів тестування, які широко застосовуються у веброзробці, такі як функціональне, інтерфейсне, рольове, інтеграційне, автоматизоване та інші [37].

Функціональне тестування — це вид тестування, який перевіряє, чи відповідає система функціональним вимогам, зазначеним у технічному завданні. Його мета упевнитися, що кожна функція працює відповідно до очікуваного сценарію. Наприклад, це перевірка входу користувача з правильними та неправильними даними, поселення студента в зайняту кімнату.

Тестування інтерфейсу користувача фокусується на зовнішньому вигляді й поведінці інтерфейсу. Це тестування перевіряє, чи всі елементи правильно відображаються, чи працюють кнопки, поля, таблиці, мапа. Метою такого тестування є забезпеченості логічністі і зручністі взаємодії з інтерфейсом. Наприклад, це чи видається повідомлення про успішне збереження даних.

Тестування з ролями передбачає перевірку рівнів доступу, щоб переконатися, що студент не бачить панель декана, відповідальна особа не може редагувати чужі гуртожитки та адміністратор має повний доступ до рішень по актах. Це важливо для захисту даних та дотримання політик доступу.

Інтеграційне тестування перевіряє взаємодії між модулями. Наприклад, клієнт React та база даних MySQL. Важливо впевнитися, що сервер обробляє запити, відповідає з коректними статусами, а клієнт правильно реагує на помилки або успіх.

Кожен із вказаних видів тестування доповнює один одного та дозволяє всебічно оцінити стабільність системи. Враховуючи, що клієнт реалізований у вигляді SPA на базі React, особливу увагу приділялось динамічному оновленню

даних, роботі з маршрутизацією, обробці подій, формам, мапі гуртожитку та фільтраційним таблицям.

Було проведено ручне тестування системи, з конкретними сценаріями та результати наведено в таблиці 4.1.

Таблиця 4.1 — Результати ручного тестування розроблюваної системи

Модуль	**Сценарій**	**Результат**
Авторизація	Вхід, збереження токена, маршрутизація по сторінках	Успішно
Ролі користувачів	Відображення функцій за роллю	Успішно
Список студентів	Завантаження, пошук, фільтрація, перегляд профілю	Успішно
Форма поселення	Валідація, вибір кімнати, обробка обмежень	Успішно
Мапа гуртожитку	Завантаження кімнат, кольори, інформація по кліку	Успішно
Сторінка актів	Створення, перегляд, прикріплення файлу	Успішно
Акти	Прийняття/відхилення, перегляд документів	Успішно
Рейтинг студентів	Додавання балів, автоматичне оновлення списку	Успішно
Робота з Google Drive	Завантаження файлу, генерація посилання, перегляд	Успішно
Звіти та аналітика	Фільтрація, експорт, вивід таблиць	Успішно

Під час тестування були виявлені та усунуті кілька недоліків. Зокрема, було виправлено некоректне оновлення інформації про кімнату після переселення шляхом додавання повторного запиту та використання useEffect. Також усунуто можливість подання порожнього акту — для цього було впроваджено валідацію. Крім того, було обмежено доступ студента до адміністративних маршрутів шляхом реалізації перевірки ролі на клієнтській стороні.

Отже, тестування клієнтської частини підтвердило її стабільну роботу, відповідність функціональним вимогам та зручність у користуванні.

4.2 Перевірка продуктивності онлайн системи контролю проживання студентів у гуртожитках

Основною метою оцінки продуктивності є визначення, наскільки швидко,

ефективно та стабільно працює клієнтська частина системи під навантаженням, з великою кількістю даних або у складних сценаріях. Наприклад, при фільтрації великих таблиць або швидкому перемиканні між вкладками.

Також важливо оцінити час завантаження, реакцію на дії користувача та відповідь інтерфейсу у типових сценаріях роботи декана, адміністратора та відповідальної особи гуртожитку.

Тестування продуктивності проводилося за допомогою інструментів Chrome DevTools, React DevTools та Lighthouse.

Chrome DevTools — це набір інструментів веб-розробника, вбудованих безпосередньо в браузер Google Chrome. DevTools дає змогу оперативно редагувати сторінки та швидко діагностувати проблеми, що допомагає швидше створювати якісніші веб-сайти. Використано інструменти Performance, Network для замірів часу відображення компонентів, часу відповіді API, затримок інтерфейсу [38].

Інструмент Network показав що елементи та зображення завантажуються без затримок та швидко. Результати тестування інструментом Network наведено на рисунку 4.1. В колонці Time зображено час завантаження завантаження компонентів. Видно, що цей результат не перевищує 1000 мс, що свідчить про високу швидкість завантаження елементів.

Інструмент Performance надав інформацію, що самий складний елемент завантажується за 0.33, що свідчить про швидке та оптимізоване відображення елементів. Сукупний зсув макету становить 0, що означає коректність відображення елементів. Результати тестування інструменту Performance наведені на рисунку 4.2.

Lighthouse — це автоматизований інструмент з відкритим вихідним кодом, який допоможе поліпшити якість веб-сторінок. Має можливість запуску на будь-якій веб-сторінці, загальнодоступній або такій, що вимагає автентифікації. Інструмент проводить аудит продуктивності, доступності, прогресивних веб-додатків, SEO та багато чого іншого [39].

Результати тестування головної сторінки з інтерактивною мапою наведено на рисунку 4.3. Загальна оцінка продуктивності становить 82, що входить в проміжок

50-89 і є задовільним показником, зважаючи на складність представленого інтерфейсу. Через наявність складних об’єктів на сторінці та інтерактивної мапи, відбувається довше завантаження сторінки.

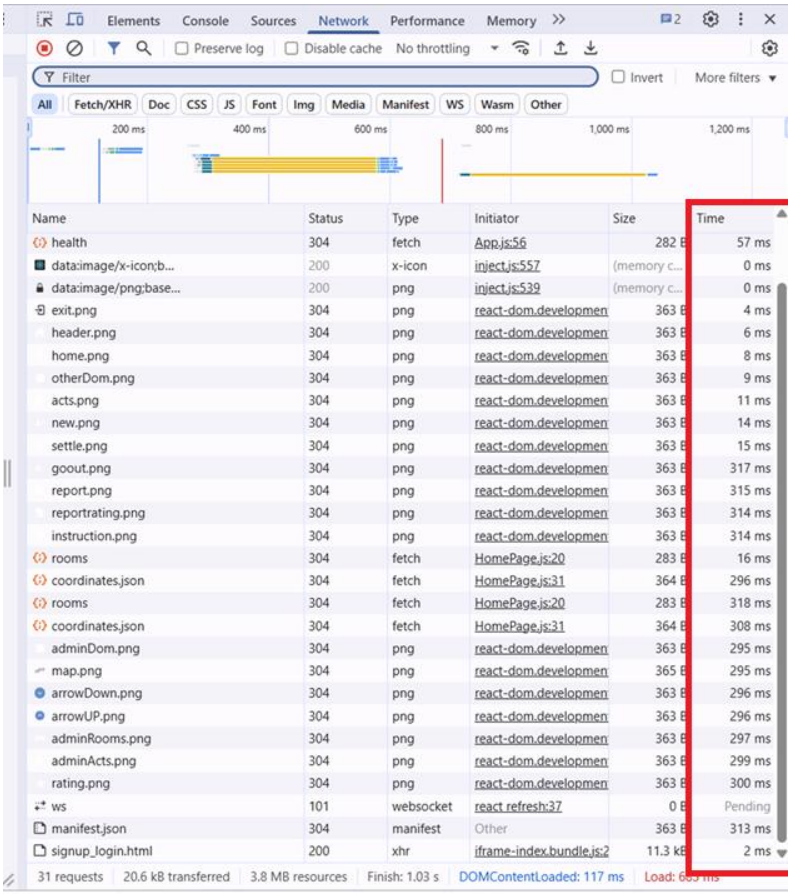

Рисунок 4.1 — Результати тестування часу завантаження елементів в Network

Рисунок 4.2 — Результати тестування інструменту часу завантаження елементів в Performance

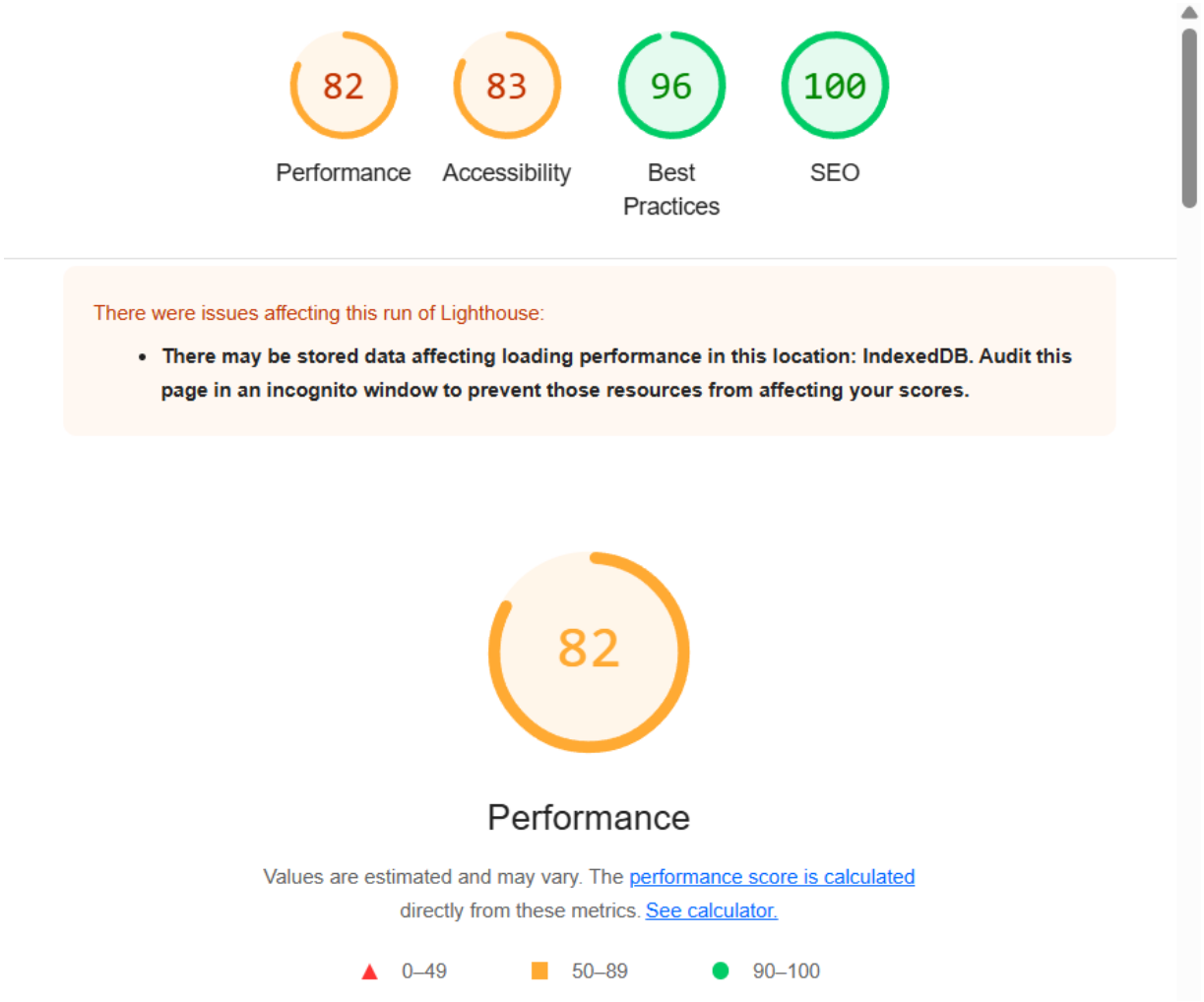

Рисунок 4.3 — Результати тестування головної сторінки з інтерактивною мапою

Для додаткових результатів роботи системи перевіримо інші модулі, наприклад модуль «Акти». Результати тестування модулю «Акти» наведено на рисунку 4.4. Показник продуктивності для цього модуля становить 95, що входить в проміжок 90-100 і свідчить про високу швидкість завантаження та оптимізовану роботу інтерфейсу. Це підтверджує, що модуль функціонує стабільно, швидко реагує на дії користувача та не має критичних вузьких місць у продуктивності.

Інструмент React Profiler, який входить в розширення React DevTools, дозволяє виявляти «важкі» компоненти та надлишкові відображення елементів. Провівши тестування між переходами об’єтів, було зафіксовано час рендерингу 5,3 мілісекунди, що вказує на швидку реакцію інтерфейсу на зміну стану. Окрім цього, Layout-ефекти зайняли лише 0,2 мс, а пасивні ефекти – 0,3 мс, що свідчить про оптимізовану логіку компонентів без зайвих навантажень при зміні сторінки.

Результати тестування швидкості рендерингу компонентів в React Profiler наведено на рисунку 4.5.

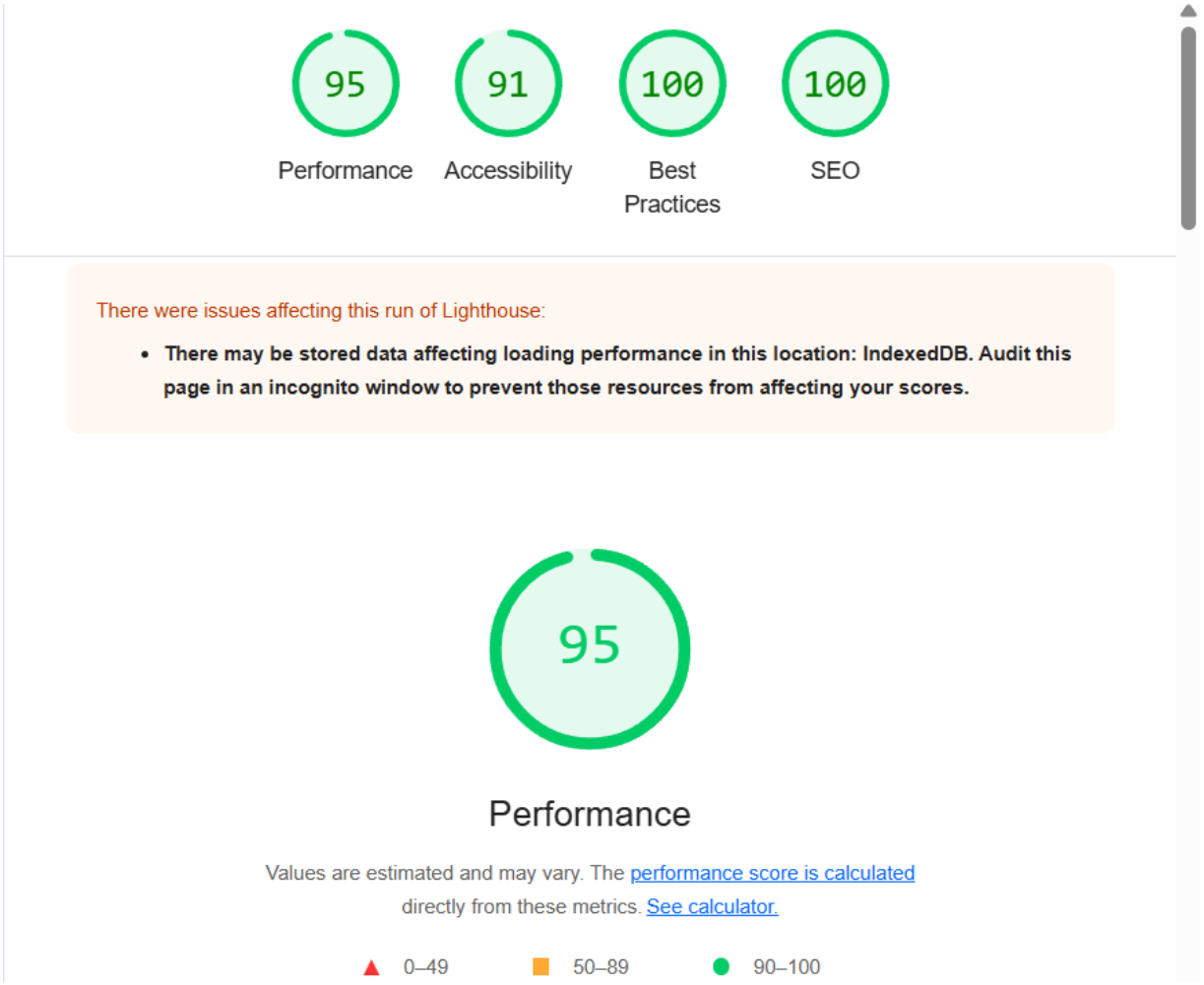

Рисунок 4.4 — Результати тестування модулю «Акти»

Результати тестування продуктивності онлайн системи наведено в таблиці 4.2.

Також було оптимізовано React-компоненти. Компоненти розділено на дрібні частини для запобігання зайвим перерендерам. Створено віртуалізацію списків через map() з обмеженням виводу при великій кількості студентів.

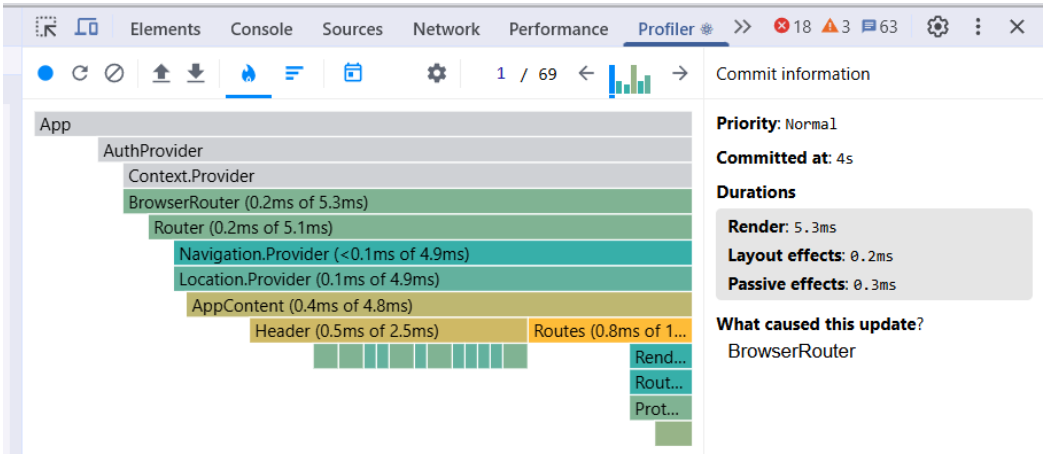

Рисунок 4.5 — Результати тестування інструментом React Profiler

Таблиця 4.2 — Результати тестування продуктивності онлайн системи

Метрика	**Значення**
Час початкового завантаження	~0.33 с
Час відповіді API	100—200 мс
Завантаження мапи гуртожитку	< 1 с
Час відкриття форми актів	< 500 мс
Ресурси при 10 одночасних сесіях	стабільно, без затримок

Великі компоненти, наприклад, ActsPage завантажуються тільки при потребі.

Усі запити реалізовано через axiosInstance з централізованою обробкою помилок. Результати деяких запитів кешуються на стороні клієнта, наприклад, список гуртожитків. Повторне завантаження даних відбувається лише за потреби, наприклад при оновленні, а не при кожному перегляді.

Інтерактивна мапа реалізована у вигляді SVG з мінімальною кількістю DOM-елементів. Стани компонентів розділено так, щоб оновлення однієї кімнати не викликало відображення всіх поверхів.

Отже, після оптимізації клієнтська частина системи демонструє стабільну роботу як на середньо статичних, так і на низько потужних пристроях, забезпечуючи швидкий доступ, плавну навігацію та низьку затримку при обміні даними з сервером.

4.4 Розгортання розробленої онлайн системи контролю проживання студентів у гуртожитках

Система була розгорнута у тестовому режимі на ресурсах Netlify, Render.com та Railway.app. Було симульовано повноцінний цикл взаємодії між ними, а саме поселення, переселення, фіксація порушень, додавання актів та формування рейтингу.

Клієнтська частина системи, була розгорнута на платформі Netlify. Цей хостинг забезпечує безкоштовне розгортання додатків та підтримує автоматичне оновлення при кожній зміни в репозиторії Github. Також Netlify надає швидке кешування ресурсів, підтримку HTTPS за замовчуванням і дозволяє забезпечити стабільну роботу інтерфейсу.

Серверну частину розгорнуто на хмарній платформі Render.com. Цей сервіс надає середовище для хостингу Express-серверів із підтримкою автоматичного перезапуску та логування. Проте Render має ряд обмежень у безкоштовному тарифному плані. Ліміт 500 МБ оперативної пам’яті, що змушує оптимізувати використання ресурсів. Також сервер зупиняється після 15 хвилин бездіяльності, і при наступному запиті потребує час на включення, що може викликати короткі затримки.

Для зберігання даних використано хмарну MySQL-базу даних на платформі Railway.app. Платформа забезпечує простий інтерфейс для створення, міграцій і резервного копіювання таблиць. Основні таблиці були перенесені на Railway з локального середовища за допомогою SQL. В безкоштовному тарифному плані доступний ліміт у 500 МБ сховища, що потребує регулярного моніторингу й видалення застарілих записів.

Для збереження працездатності системи було реалізовано ручну оптимізацію запитів, обмеження обсягу завантажуваних файлів та, за потреби, кешування відповідей.

Під час перевірки встановлено, що створення акту про порушення вимагало до 10-15 хвилин через необхідність ручного оформлення та реєстрації. Завдяки використанню системи цей процес скорочено до 2-3 хвилин, з повним введенням

інформації. Аналогічно, пошук інформації про конкретного студента, його кімнату або рейтинг, що раніше займав 5-10 хвилин, тепер виконується менш ніж за 5 секунд. А також надало можливість мобільності, наприклад можна уточнити інформацію по студенту з телефону, не дзвонячи в деканат. Створення звітів, що раніше потребувало до півгодини, зараз реалізується практично миттєво, протягом 5-10 секунд.

Швидкість обробки основних дій зросла в середньому в 5-10 разів.

До впровадження системи часто виникали помилки через дублювання інформації, неузгодженість записів і втрату файлів, які зберігались локально. Після впровадження всі дані зберігаються централізовано у єдиній базі з вбудованими механізмами перевірки коректності, а кожен акт або документ жорстко закріплюється за відповідним записом у базі. Завдяки інтеграції з Google Drive забезпечено постійний доступ до повного комплекту супровідних матеріалів, що формують доказову базу для прийняття рішень.

Усі дані про студентів, кімнати, переселення та акти об'єднані в єдину інформаційну систему, що дозволяє здійснювати глибокий аналіз змін, контролювати динаміку проживання, об’єктивно оцінювати поведінку студентів через рейтингування. Модуль рейтингу, зокрема, надає можливість відслідковувати зміни балів у часі та формувати звітність за окремими категоріями.

На сайті факультету інформаційних технологій та комп’ютерної інженерії (ФІТКІ) Вінницького національного технічного університету розміщено посилання, при переході за яким відбувається перенаправлення на вебсторінку розробленого вебзастосунку, що розгорнутий на платформі Netlify. Аналогічно реалізовано окреме посилання для модуля реєстрації порушень правил проживання студентів у гуртожитку. На рисунку 4.6 наведено вигляд вебсторінки «Gurt Systems» на сайті ФІТКІ з вказаними посиланнями.

Отже, використання системи створює передумови для якісного цифрового трансформування управління студентським житлом. Онлайн систем контролю проживання студентів у гуртожитках дозволяє зменшити навантаження на

адміністрацію, уникати помилок, пришвидшити час виконання звітності та зробити всі необхідні дані доступними в кілька кліків.

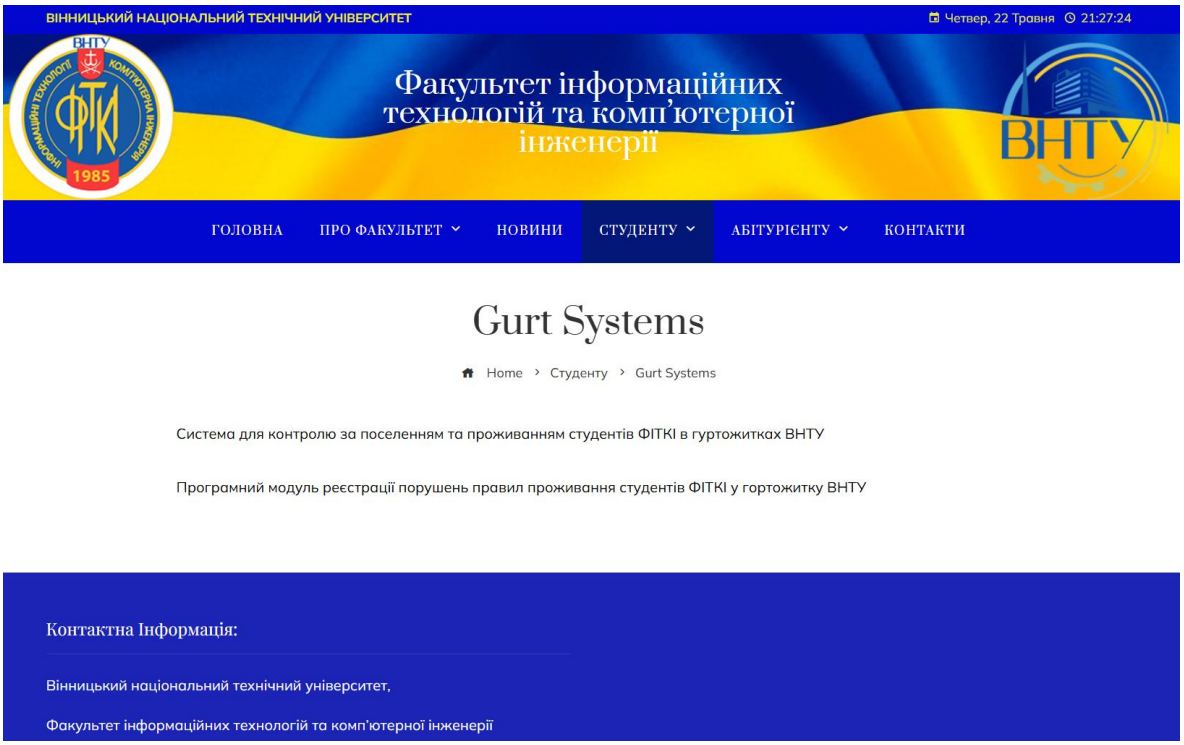

Рисунок 4.6 — Вигляд вебсторінки на сайті ФІТКІ з розміщеним посиланням на розроблену систему

4.5 Перспективи подальшого розвитку та масштабування

Розширення можливостей системи обліку проживання студентів у гуртожитках є наступним етапом після впровадження. На поточному етапі реалізації видно потенціал для вдосконалення функціональності, покращення взаємодії з користувачами та підвищення рівня автоматизації.

Одним із основних напрямів розвитку є інтеграція з іншими інформаційними системами навчального закладу, наприклад JetIQ. Це дозволить автоматично підтягувати інформацію про студентів, їх академічний статус, пільги, академічну заборгованість та інші релевантні параметри, що впливають на поселення.

Ще одним важливим кроком у розвитку є розширення функціональності аналітики. В перспективі можна реалізувати систему інформаційних блоків для

адміністрації, де будуть відображатися статистичні дані по заповненості кімнат, кількості порушень, середньому рейтингу по групах або гуртожитках.

З технічної точки зору система вже має модульну архітектуру, що дозволяє масштабувати її на інші гуртожитки, факультети або навіть навчальні заклади.

Ще один можливий напрям розвитку це мобільна версія застосунку. Хоча вебверсія адаптована для мобільних пристроїв, створення окремого застосунку на базі React Native дозволить впровадити push-сповіщення для студентів, наприклад, про прийнятий акт або оновлення рейтингу, а також спростити доступ до мапи та документів.

Отже, створена система є не лише готовим програмним рішенням для автоматизації обліку студентів у гуртожитках, але і надійною основою для подальшого технологічного розвитку освітньої інфраструктури.

Оцінка продуктивності продемонструвала хороші результати завдяки застосуванню оптимізацій на рівні інтерфейсу та обробки запитів, що дозволило досягти швидкого відгуку системи навіть при роботі з великим обсягом інформації. Інтеграція з Google Drive дала змогу реалізувати централізоване та безпечне зберігання супровідних документів, що підвищує прозорість та обґрунтованість адміністративних рішень.

Отже, можна зробити висновок про успішну реалізацію етапу тестування та готовність системи до продуктивного використання в закладах вищої освіти.

ВИСНОВКИ

В рамках виконання бакалаврської кваліфікаційної роботи реалізовано комп’ютерну онлайн систему обліку, рейтингування та контролю проживання студентів у гуртожитку, яка забезпечує автоматизацію ключових адміністративних процесів у сфері житлового забезпечення студентів. Розроблена система розширяє функціональні можливості наявної системи обліку для гуртожитку та дає можливість формування рейтингу, контролю дисциплінарних заходів, ведення звітності за окремим студентом та збереження супровідних документів.

Під час аналітичного дослідження визначено основні проблеми сучасного стану обліку студентів у гуртожитках, а саме значний рівень залежності від ручного ведення документації, недостатній рівень прозорості процедур, низька швидкість формування звітності та високий ризик втрати інформації.

Проведений огляд та порівняльний аналіз наявних аналогів показав, що жодна з існуючих систем не задовольняє специфічних потреб адміністрації гуртожитків, таких як підтримка інтерактивної мапи, автоматичне рейтингування, централізована робота з актами порушень та заохочень, а також інтеграція з хмарним сховищем.

На основі визначених вимог розроблено архітектуру онлайн системи контролю проживання студентів у гуртожитках, реалізованої у вигляді вебзастосунку, що включає клієнтську частину на базі React.js, серверну частину на Node.js з використанням Express, реляційну базу даних MySQL та Google Drive як середовище для зберігання супровідних документів.

Також, проведено тестування системи, що показало стабільну роботу інтерфейсу, правильну обробку запитів, високу швидкість завантаження сторінок та ефективну взаємодію з базою даних.

Впровадження системи в тестовому режимі сприяло помітному зменшенню часу, необхідного для обробки звернень студентів, автоматизувати процедури поселення та переселення, підвищити точність формування рейтингу і прозорість ухвалення рішень адміністрацією.

Посилання, за яким можна перейти на розроблену комп’ютерну онлайн

систему обліку, рейтингування та контролю проживання студентів у гуртожитку розміщено на сайті факультету ІТКІ ВНТУ.

Система має можливість масштабування для використання в кількох гуртожитках або навіть університетах, а також інтеграцією з іншими інформаційними системами навчальних закладів. Додаткові можливості розвитку включають створення мобільного застосунку та розширення можлиовстей.

Результати виконання свідчать про практичну значущість, ефективність і доцільність впровадження розробленої системи у навчальних закладах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Онлайн система обліку, рейтингування та контролю проживання студентів у гуртожитку // Маліцький В. В., Войцеховська О. В. Матеріали LIV Всеукраїнської науково-технічної конференції підрозділів Вінницького національного технічного університету (Вінниця, 2025 р.) [Електронний ресурс]. Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2025/paper/view/23812/19633

2. Свідоцтво про реєстрацію авторського права на твір № 135686. Комп'ютерна програма «Програмний модуль реєстрації порушень правил проживання студентів у гуртожитку» / Бабюк Н. П., Войцеховська О. В., Маліцький В. В. Дата реєстрації 05.05.2025 р.

3. Система управління навчанням: вебсайт. URL: https://uk.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D1%96%D0%BD%D0%BD%D1%8F_%D0%BD%D0%B0%D0%B2%D1%87%D0%B0%D0%BD%D0%BD%D1%8F%D0%BC (дата звернення: 07.03.2025)

4. Інформаційно-виробнича система «Освіта» 2025 : вебсайт. URL: https://osvita.net/ua/ (дата звернення: 08.03.2025)

5. АС «Студмістечко»: вебсайт. URL: https://vuz.osvita.net/as-studmistechko/ (дата звернення: 08.03.2025)

6. Квєтний Р. Н. , Паламарчук Є. А. , Бісікало О. В. , Коваленко О. О.. Концепція сучасного університету на основі інструментів електронної екосистеми управління освітніми процесами JETIQ ВНТУ . Концепція сучасного університету на основі інструментів електронної екосистеми управління освітніми процесами JETIQ ВНТУ [Електронний ресурс] : наукова доповідь загальним зборам НАПН України «Науково-методичне забезпечення цифровізації освіти України: стан, проблеми, перспективи», 18 листопада 2022 р. / Р. Н. Квєтний, Є. А. Паламарчук, О. В. Бісікало, о. О. Коваленко // Вісник Національної академії педагогічних наук України. – Електрон. текст. дані. – 2022. – Т. 4, № 2. – Режим доступу:

https://visnyk.naps.gov.ua/index.php/journal/article/view/314. (дата звернення: 09.03.2025)

7. Microsoft Excel : вебсайт. URL: https://www.microsoft.com/uk-ua/microsoft-365/excel (дата звернення: 10.03.2025)

8. Основні типи архітектури програмного забезпечення : вебсайт. URL: https://www.artofba.com/uk/post/main-types-of-software-architecture (дата звернення: 16.03.2025)

9. Software Architecture in Practice, 3rd Edition, Len Bass, Paul Clements, Rick Kazman, 2013 (дата звернення: 16.03.2025)

10. Learn Angular: вебсайт. URL: https://angular.dev/ (дата звернення: 20.03.2025)

11. React. The library for web and native user interfaces : вебсайт. URL: https://react.dev/ (дата звернення: 21.03.2025)

12. Vue.js - The Progressive JavaScript Framework | Vue.js : вебсайт. URL: https://vuejs.org/ (дата звернення: 22.03.2025)

13. Django: The web framework for perfectionists with deadlines : вебсайт. URL: https://www.djangoproject.com/ (дата звернення: 23.03.2025)

14. Laravel - The PHP Framework For Web Artisans : вебсайт. URL: https://laravel.com/ (дата звернення: 24.03.2025)

15. Run JavaScript Everywhere : вебсайт. URL: https://nodejs.org/en (дата звернення: 25.03.2025)

16. npm Docs | Documentation for the npm registry, website, and command-line interface: вебсайт. URL: https://docs.npmjs.com/ (дата звернення: 25.03.2025)

17. Express Fast, unopinionated, minimalist web framework for Node.js : вебсайт. URL: https://expressjs.com/ (дата звернення: 26.03.2025)

18. Cloud Computing Services - Amazon Web Services (AWS) : вебсайт. URL: https://aws.amazon.com/ (дата звернення: 28.03.2025)

19. Microsoft Azure : вебсайт. URL: https://azure.microsoft.com/ (дата звернення: 28.03.2025)

20. Microsoft OneDrive Dev Center | APIs and app development : вебсайт.

URL: https://developer.microsoft.com/en-us/onedrive (дата звернення: 29.03.2025)

21. Google Cloud Console : вебсайт. URL: https://console.cloud.google.com/ (дата звернення: 29.03.2025)

22. Google Drive API overview : вебсайт. URL: https://developers.google.com/workspace/drive/api/guides/about-sdk (дата звернення: 29.03.2025)

23. MySQL : вебсайт. URL: https://www.mysql.com/ (дата звернення: 30.03.2025)

24. Firebase Authentication - Google: вебсайт. URL: https://firebase.google.com/docs/auth (дата звернення: 01.04.2025)

25. Visual Studio Code - Code Editing. Redefined : вебсайт. URL: https://code.visualstudio.com/ (дата звернення: 01.04.2025)

26.Postman: The World's Leading API Platform | Sign Up for Free : вебсайт. URL: https://www.postman.com/ (дата звернення: 03.04.2025)

27. GitHub : вебсайт. URL: https://github.com/GIT (дата звернення: 03.04.2025)

28. MySQL Workbench : вебсайт. URL: https://www.mysql.com/products/workbench/ (дата звернення: 04.04.2025)

29. Як працює SPA (Single Page Application) — усе, що потрібно знати : вебсайт. URL: https://dou.ua/forums/topic/50894/ (дата звернення: 07.04.2025)

30. Overview of ASP.NET Core MVC : вебсайт. URL: https://learn.microsoft.com/uk-ua/aspnet/core/mvc/overview?view=aspnetcore-9.0 (дата звернення: 08.04.2025)

31. REST API як спосіб спілкування компонент веб-додатків : вебсайт. URL: https://foxminded.ua/shcho-take-rest-api/ (дата звернення: 08.04.2025)

32. Що таке ендпоінти (endpoints) : вебсайт. URL: https://pmtips.com.ua/qna/shcho-take-endpointy (дата звернення: 12.04.2025)

33. JSON Web Token (JWT) Debugger : вебсайт. URL: https://jwt.io/ (дата звернення: 15.04.2025)

34. Google API Developer Console : вебсайт. URL:

https://console.cloud.google.com/apis/dashboard?inv=1&invt=AbwrAw&project=gurtsystems (дата звернення: 16.04.2025)

35. Різниця між HTTP та HTTPs : вебсайт. URL: https://hostiq.ua/wiki/ukr/http-https/ (дата звернення: 17.04.2025)

36. Що таке Middleware? : вебсайт. URL: https://pnn.com.ua/ua/blog/detail/what-is-middleware (дата звернення: 18.04.2025)

37. What is Axios? : вебсайт. URL: https://axios-http.com/docs/intro (дата звернення: 19.04.2025)

38. «Введення в тестування програмного забезпечення», Луїза Тампе, 2018 (дата звернення: 22.04.2025)

39. Chrome DevTools | Chrome for Developers : вебсайт. URL: https://developer.chrome.com/docs/devtools (дата звернення: 24.04.2025)

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.

21 березня 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання бакалаврської кваліфікаційної роботи

«Комп'ютерна онлайн система обліку, рейтингування та контролю проживання студентів у гуртожитку»

08-54.БКР.013.00.000 ПЗ

Науковий керівник: доцент к.т.н.

_____Войцеховська О.В.

Студент групи 1КІ-23мс

_____Маліцький В.В.

м. Вінниця — 2025

1 Підстава для використання бакалаврської кваліфікаційної роботи (БКР)

1.1У сучасному світі, де цифровізації освітніх процесів приділяють особливу увагу, важливим є використанням нових технологій. Однією з ключових сфер, яка потребує систематизації та оптимізації, є управління проживанням студентів у гуртожитках.

1.2 Наказ про затвердження теми кваліфікаційної роботи.

2 Мета БКР і призначення розробки

2.1 Мета роботи – розширення функціональних можливостей наявної системи обліку для гуртожитку шляхом створення комп'ютерної онлайн системи, яка забезпечить можливість формування рейтингу, контролю дисциплінарних заходів, ведення звітності за окремим студентом та збереження супровідних документів.

2.2 Призначення розробки — онлайн система обліку, рейтингування та контролю проживання студентів у гуртожитку.

3 Вихідні дані для виконання БКР

3.1 Проведення аналізу існуючих систем та технологій обліку, рейтингування та контролю проживання студентів у гуртожитку;

3.2 Розробка структури та функціональної схеми програмно- апаратного комплексу для онлайн системи обліку, рейтингування та контролю проживання студентів у гуртожитку;

3.3 Розробка клієнтської частини веб-додатку;

3.4 Розробка серверної частини веб-додатку.

3.5 Тестування клієнтської частини веб- додатку;

3.6 Впровадження онлайн системи контролю проживання студентів у гуртожитках;

4 Вимоги до виконання БКР

Головна вимога – створити систему, яка матиме практичне застосування в

контролі проживання студентів у гуртожитках.

5 Етапи БКР та очікувані результати

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

Таблиця А.1 — Етапи БКР

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Огляд і аналіз сучасного стану розвитку галузі розробки онлайн систем контролю проживання студентів у гуртожитках	21.03.25	30.03.25	Аналітичний огляд літературних джерел
2	Проєктування онлайн системи контролю проживання студентів у гуртожитках	21.03.25	30.03.25	Розділ 2
3	Програмна реалізація онлайн системи контролю проживання студентів у гуртожитках	31.03.25	13.04.25	Розділ 3
4	Тестування та впровадження онлайн системи контролю проживання студентів у гуртожитках	14.04.25	27.04.25	Розділ 4
5	Апробація та впровадження результатів дослідження	28.04.25	11.05.2025	Тези доповідей
6	Оформлення пояснювальної записки, графічного матеріалу і презентації	28.04.25	11.05.2025	пояснювальна записка, графічний матеріал і/або презентація

6 Матеріали, що подаються до захисту БКР

До захисту подаються: пояснювальна записка БКР, графічні і ілюстративні матеріали, протокол попереднього захисту БКР на кафедрі, відгук наукового керівника, відгук опонента, протоколи складання державних екзаменів, анотації до БКР українською та іноземною мовами, довідка про відповідність оформлення БКР діючим вимогам.

7 Порядок контролю виконання та захисту БКР

Виконання етапів графічної та розрахункової документації БКР

контролюється науковим керівником згідно зі встановленими термінами. Захист БКР відбувається на засіданні Екзаменаційної комісії, затвердженої наказом ректора.

8 Вимоги до оформлювання та порядок виконання БКР

8.1 При оформлюванні БКР використовуються:

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

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

— методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів спеціальності 123 «Комп'ютерна інженерія» (освітня програма «Комп'ютерна інженерія») — кафедра обчислювальної техніки, ВНТУ 2023.

8.2 Порядок виконання БКР викладено в «Положення про кваліфікаційні роботи на першому (бакалаврському) рівні вищої освіти СУЯ ВНТУ-03.02.02-П.001.01:21»

ДОДАТОК Б

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: **Комп'ютерна онлайн система обліку, рейтингування та контролю проживання студентів у гуртожитку.**

Тип роботи: кваліфікаційна робота

Підрозділ: кафедра обчислювальної техніки, ФІТКІ, 1–КІ–23мс

Науковий керівник: к.т.н., проф. каф. ОТ. Войцеховська О. В.

Unicheck	
Оригінальність	**98%**
Схожість	2%

Аналіз звіту подібності

■ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**

☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Заявляю, що ознайомлений з повним звітом подібності, який був згенерований Системою щодо роботи «Технології реалізації конвеєрної збірки динамічних обчислювальних середовищ».

Автор _____ Маліцький В.В.

Опис прийнятого рішення: **допустити до захисту**

Особа, відповідальна за перевірку Войцеховська О. В.

_____ _____

(підпис) (прізвище, ініціали)

Експерт _____ _____

(за потреби) (підпис) (прізвище, ініціали, посада)

ДОДАТОК Г

ІЛЮСТРАТИВНА ЧАСТИНА

КОМП’ЮТЕРНА ОНЛАЙН СИСТЕМА ОБЛІКУ, РЕЙТИНГУВАННЯ ТА КОНТРОЛЮ ПРОЖИВАННЯ СТУДЕНТІВ У ГУРТОЖИТКУ

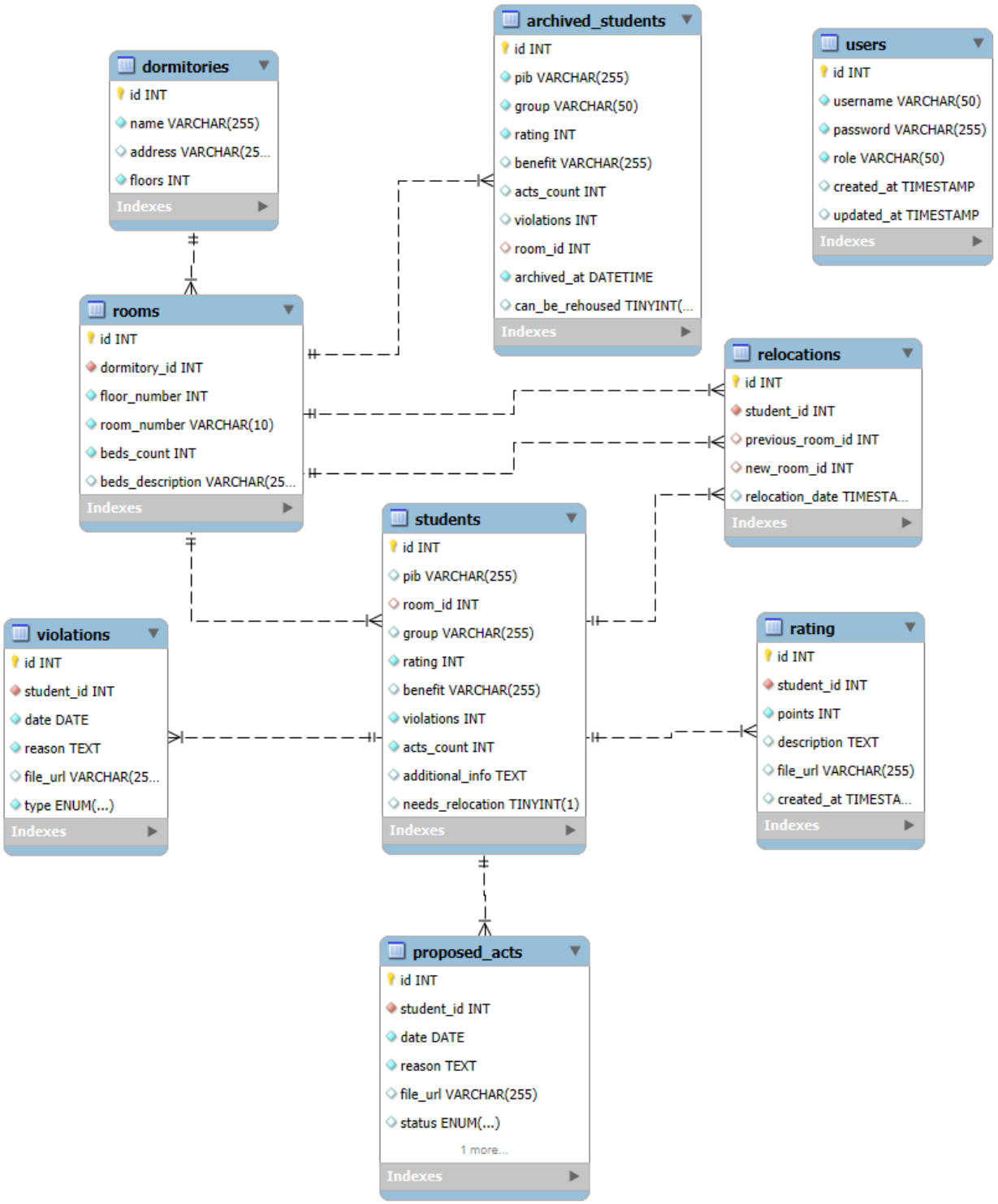

Рисунок Г.1 — ER-діаграма бази даних

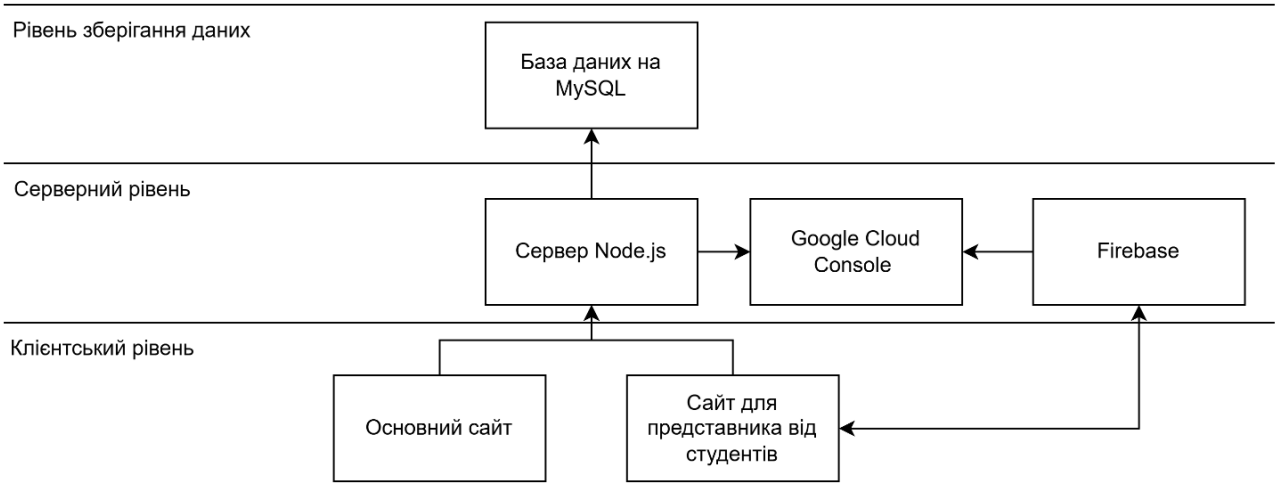

Рисунок Г.2 — Структурна схема архітектурних рівнів системи

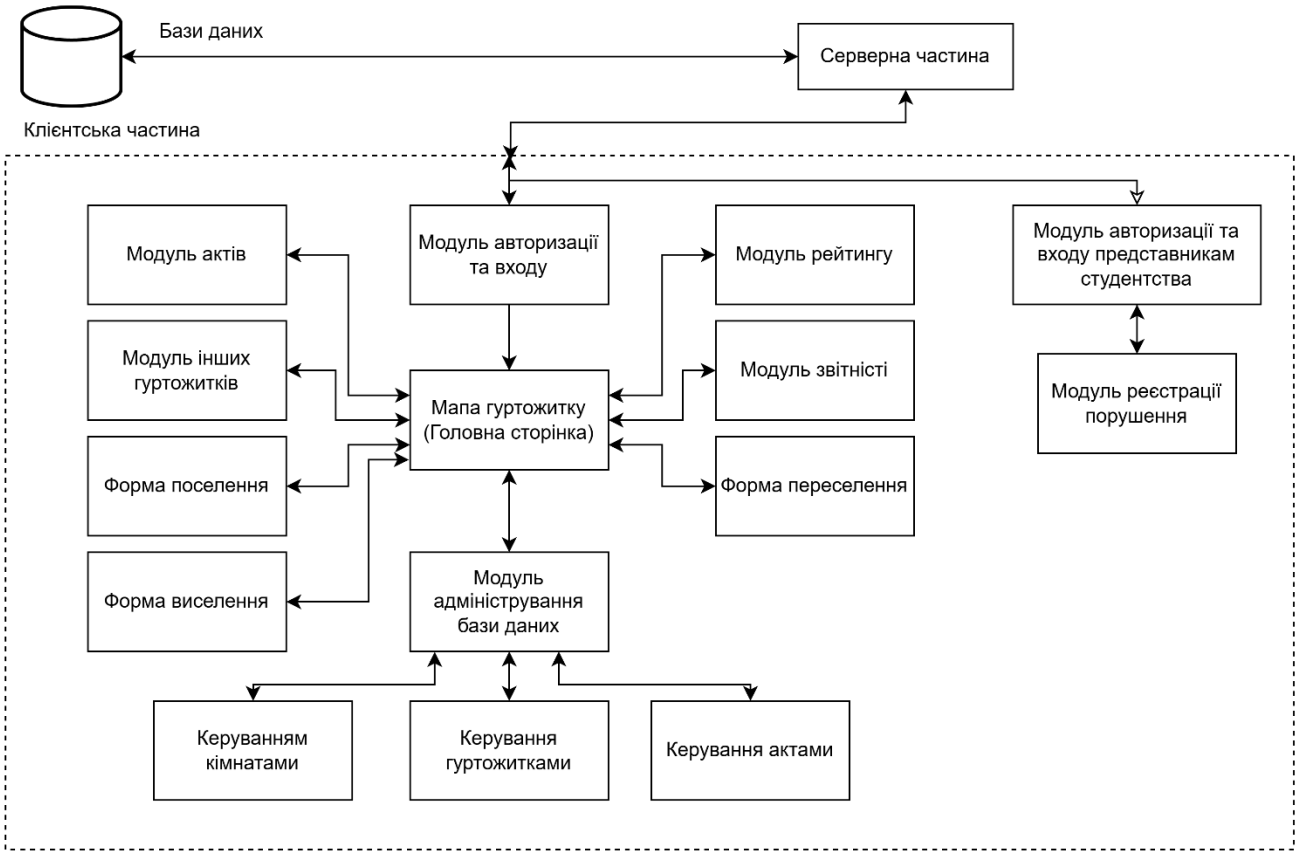

Рисунок Г.3 — Структурна схема клієнтської частини

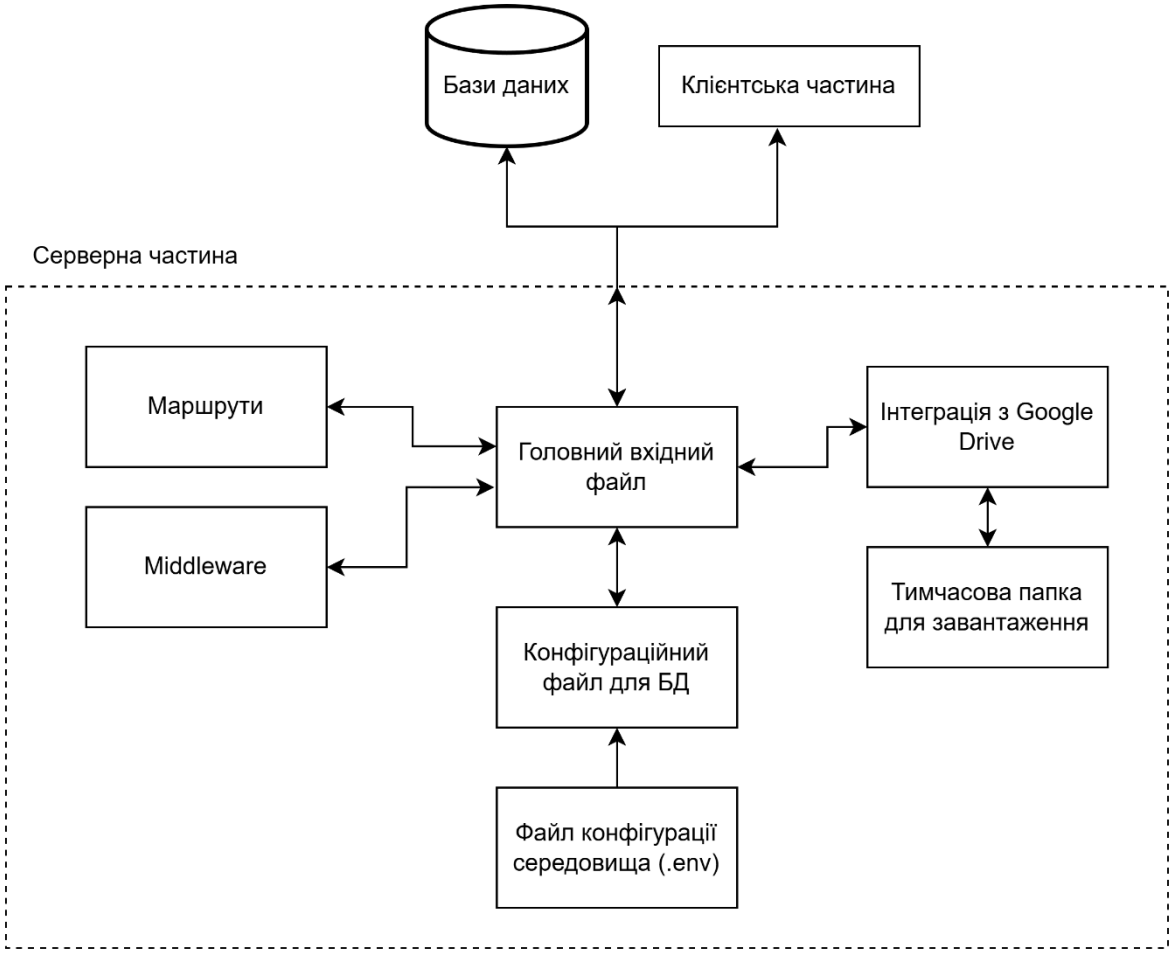

Рисунок Г.4 — Структурна схема серверної частини

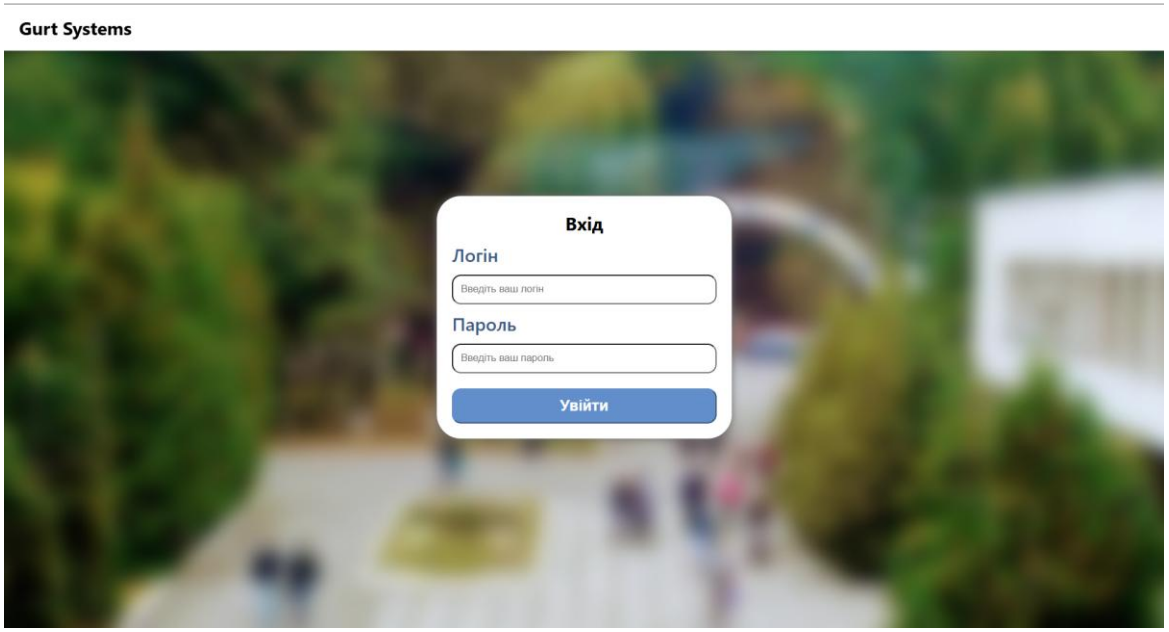

Рисунок Г.5 — Графічний інтерфейс сторінки входу системи

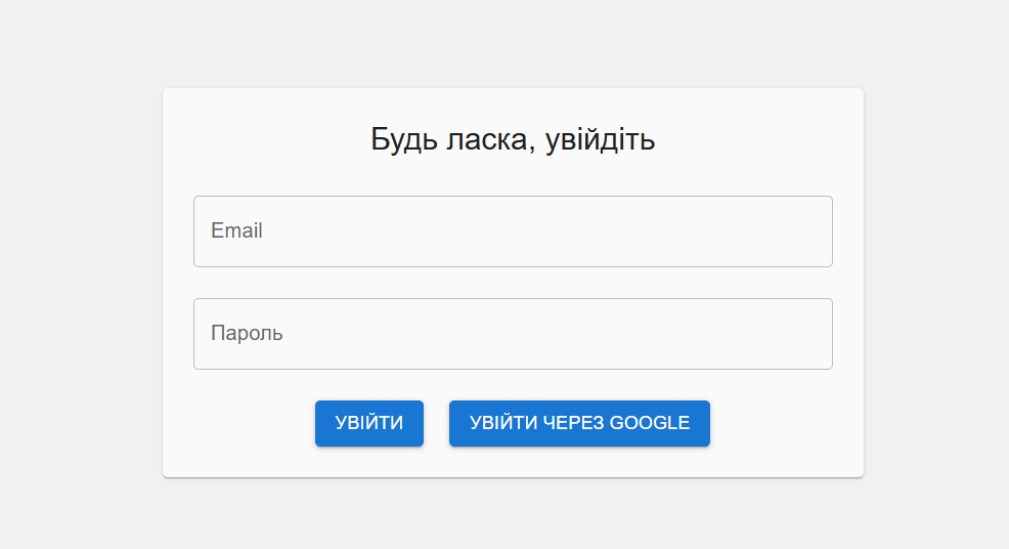

Рисунок Г.6 — Графічний інтерфейс входу в програмний модуль реєстрації порушень правил проживання студентів у гуртожитку

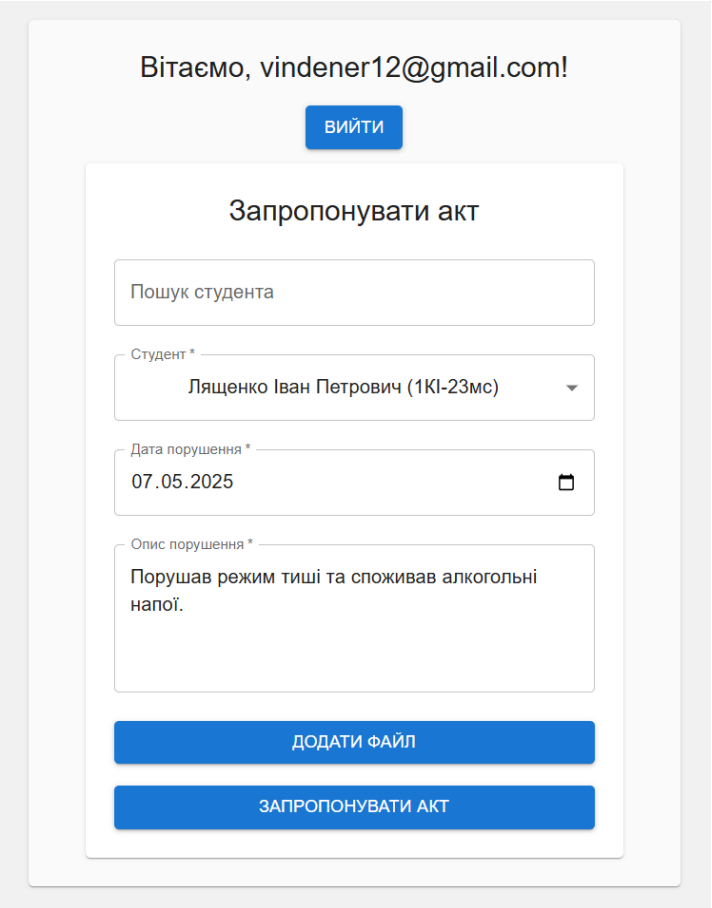

Рисунок Г.7 — Графічний інтерфейс форми подання програмного модуля реєстрації порушень правил проживання студентів у гуртожитку

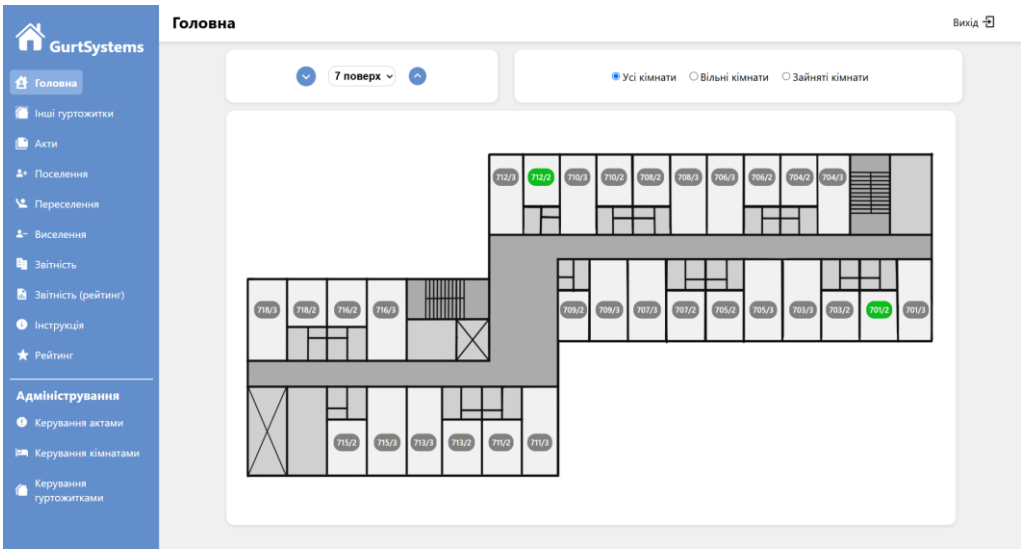

Рисунок Г.8 — Графічний інтерфейс інтерактивної мапи гуртожитку

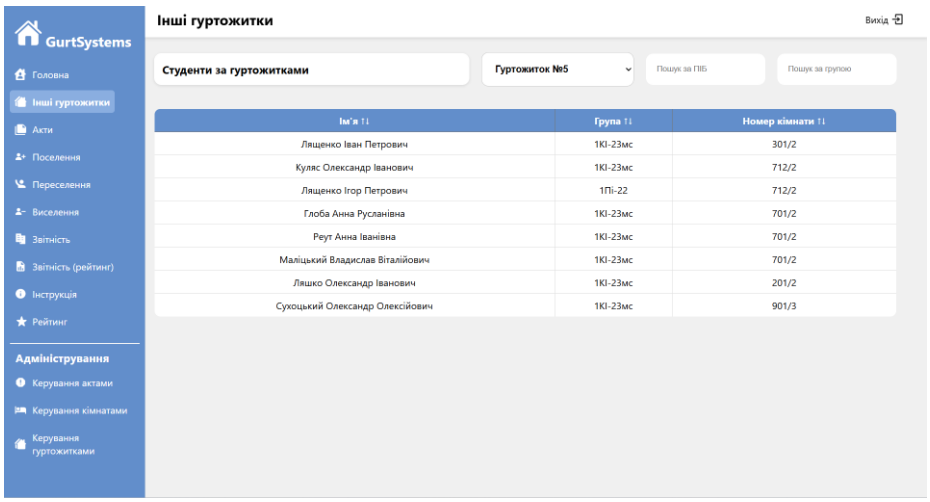

Рисунок Г.9 — Графічний інтерфейс сторінки «Інші гуртожитки»

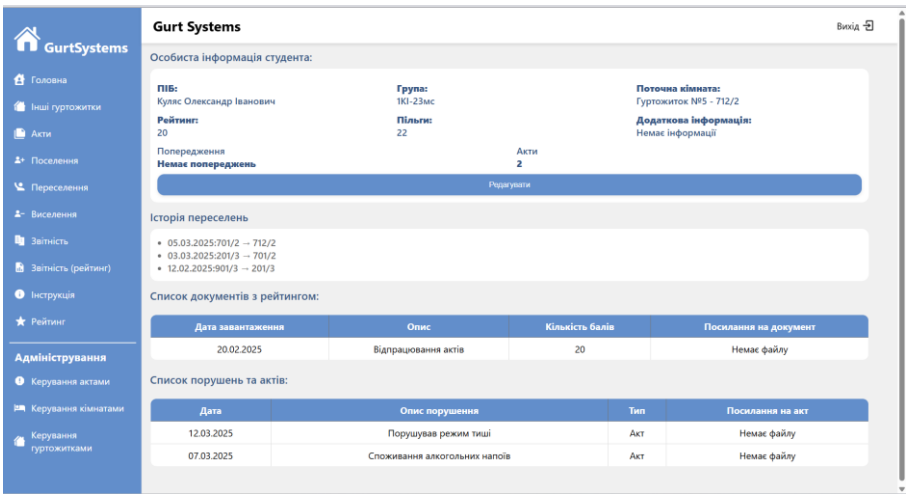

Рисунок Г.10 — Графічний інтерфейс профілю студента

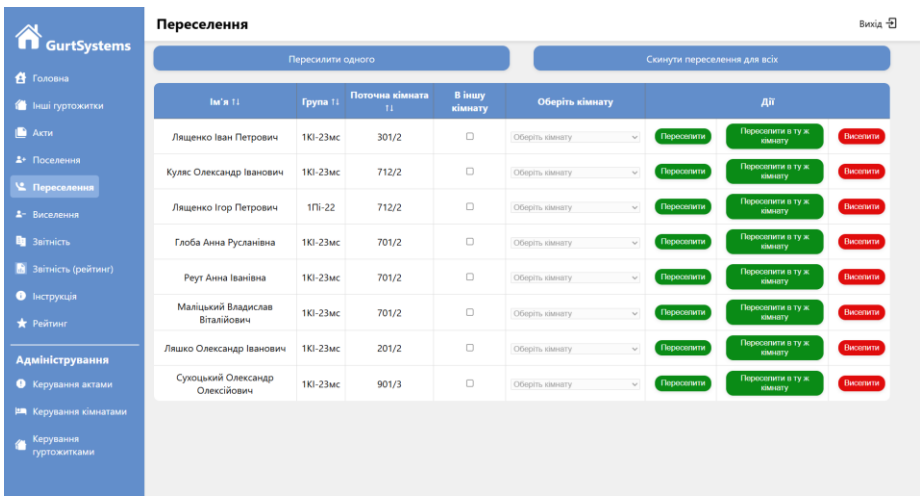

Рисунок Г.11 — Графічний інтерфейс форми переселення

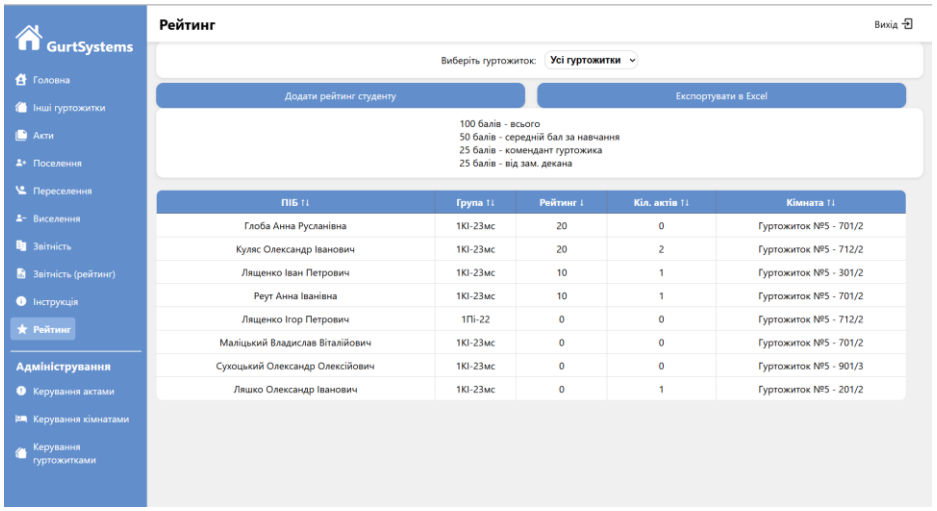

Рисунок Г.12 — Графічний інтерфейс сторінки «Рейтинг»

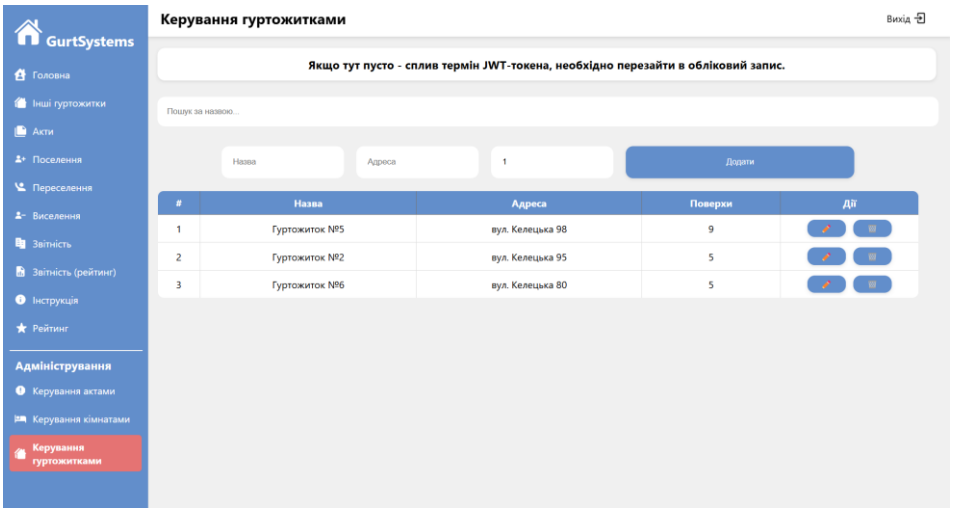

Рисунок Г.13 — Графічний інтерфейс сторінки «Керування гуртожитками»

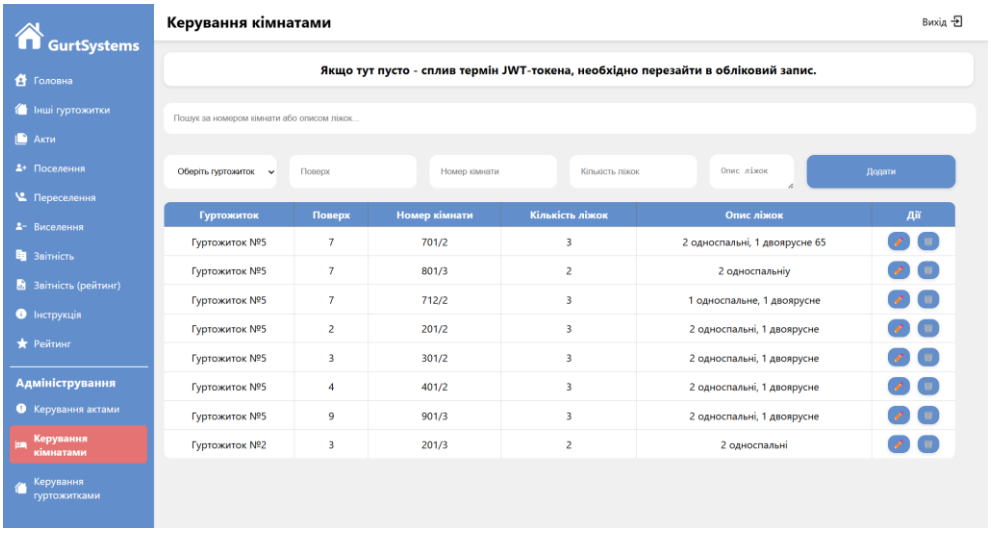

Рисунок Г.14 — Графічний інтерфейс сторінки «Керування кімнатами»

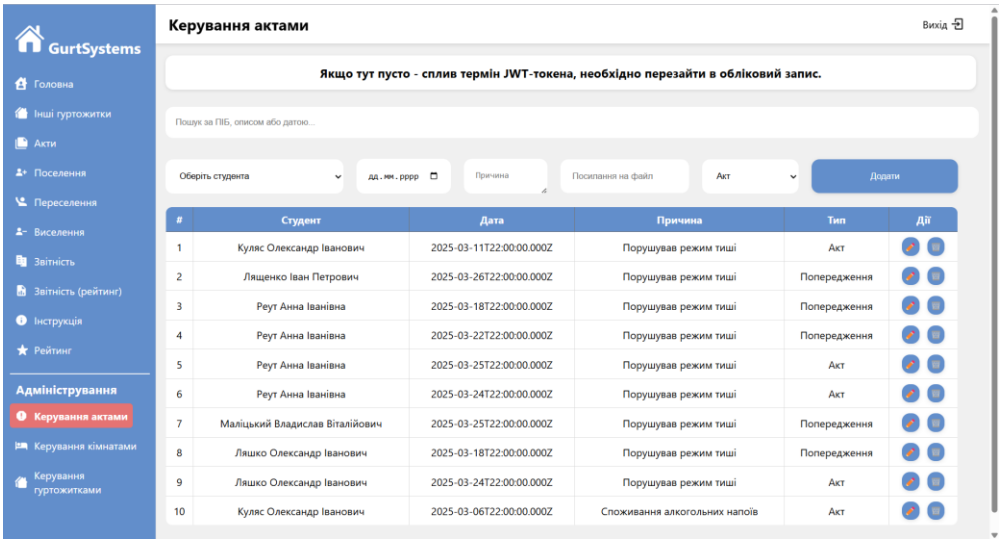

Рисунок Г.15 — Графічний інтерфейс сторінки «Керування актами»

ДОДАТОК Д

Лістинг програмного коду файлу HomePage.js

```
import { useState, useEffect } from "react";
import { useNavigate } from "react-router-dom";
import "../styles/App.css";

const HomePage = () => {
  const [selectedRoom, setSelectedRoom] = useState(null);
  const [currentFloor, setCurrentFloor] = useState(7);
  const [roomsData, setRoomsData] = useState([]);
  const [roomCoordinates, setRoomCoordinates] = useState([]);
  const [filter, setFilter] = useState("all"); // Стани фільтру: "all", "free", "occupied"
  const maxFloor = 9;
  const minFloor = 1;
  const navigate = useNavigate();
  const API = process.env.REACT_APP_API_BASE;

  useEffect(() => {
    const fetchData = async () => {
      try {
        const response = await fetch(`${API}/api/rooms`);
        const data = await response.json();
        setRoomsData(data);
      } catch (error) {
        console.error("Error fetching rooms data:", error);
      }
    };

    const fetchCoordinatesData = async () => {
      try {
        const response = await fetch("/data/coordinates.json");
        if (!response.ok) {
          throw new Error(`HTTP error! Status: ${response.status}`);
        }
        const data = await response.json();
        setRoomCoordinates(data);
      } catch (error) {
        console.error("Error loading coordinates data:", error);
      }
    };

    fetchData();
    fetchCoordinatesData();
  }, []);
  useEffect(() => {
    setSelectedRoom(null);
  }, [currentFloor]);

  const filteredRooms = Object.keys(roomCoordinates).filter((roomId) => {
    const roomFloor = parseInt(roomId.split("/")[0][0], 10);
    if (roomFloor !== currentFloor) return false;
```

```
  const room = roomsData.find((r) => r.id === roomId);

  if (filter === "free") {
    return !room || !room.occupants || room.occupants.length === 0;
  }
  if (filter === "occupied") {
    return room && room.occupants && room.occupants.length > 0;
  }
  return true; // "all"
});
const getRoomStyle = (roomId) => {
  const room = roomsData.find((r) => r.id === roomId);

  if (!room || !room.occupants || room.occupants.length === 0) {
    return { backgroundColor: "gray", color: "white" };
  }

  const maxViolations = Math.max(
    ...room.occupants.map((occupant) => occupant.violations || 0)
  );

  if (maxViolations >= 3)
    return {
      backgroundColor: "var(--full-busy)",
      color: "var(--main-color-2)",
    };
  if (maxViolations === 2)
    return {
      backgroundColor: "var(--half-busy)",
      color: "var(--main-color-2)",
    };
  if (maxViolations === 1)
    return { backgroundColor: "var(--half-busy)", color: "black" };
  return {
    backgroundColor: "var(--free-room)",
    color: "var(--main-color-2)",
  };
};
const handleRoomClick = (roomId) => {
  const room = roomsData.find((r) => r.id === roomId) || {
    id: roomId,
    beds: 0,
    occupants: [],
  };
  setSelectedRoom(room);
};

const handleFloorChange = (floor) => {
  setCurrentFloor(floor);
};
```

```
const handleStudentClick = (id) => {
  navigate(`/student/${encodeURIComponent(id)}`);
};

const countFreeBeds = (room) => {
  const occupiedBeds = room.occupants ? room.occupants.length : 0;
  return Math.max(0, room.beds - occupiedBeds);
};

const getStudentStyle = (violations) => {
  const backgroundColor =
    violations >= 3
      ? "var(--full-busy)"
      : violations === 2
      ? "var(--half-busy)"
      : violations === 1
      ? "var(--half-busy)"
      : "var(--free-room)";

  const color = backgroundColor === "var(--half-busy)" ? "black" : "white";
  return { backgroundColor, color };
};

return (
  <div className="page-container">
    <div className="controls-container">
      <div className="floor-controls">
        <img
          src={`./image/controls/arrowDown.png`}
          alt="⬇️"
          className="arrow"
          onClick={() =>
            setCurrentFloor((prev) => Math.max(minFloor, prev - 1))
          }
        />
        <select
          className="header-content-select"
          value={currentFloor}
          onChange={(e) => handleFloorChange(Number(e.target.value))}
        >
          {Array.from(
            { length: maxFloor - minFloor + 1 },
            (_, i) => minFloor + i
          ).map((floor) => (
            <option key={floor} value={floor}>
              {floor} поверх
            </option>
          ))}
        </select>
        <img
          src={`./image/controls/arrowUP.png`}
          alt="⬆️"
```

```
        className="arrow"
        onClick={() =>
          setCurrentFloor((prev) => Math.min(maxFloor, prev + 1))
        }
      />
    </div>

    <div className="filter-controls">
      <label>
        <input
          type="radio"
          name="filter"
          value="all"
          checked={filter === "all"}
          onChange={() => setFilter("all")}
        />
        Усі кімнати
      </label>
      <label>
        <input
          type="radio"
          name="filter"
          value="free"
          checked={filter === "free"}
          onChange={() => setFilter("free")}
        />
        Вільні кімнати
      </label>
      <label>
        <input
          type="radio"
          name="filter"
          value="occupied"
          checked={filter === "occupied"}
          onChange={() => setFilter("occupied")}
        />
        Зайняті кімнати
      </label>
    </div>
  </div>
  <div className="map-container">
    <img
      src={"./image/map.png"}
      alt={`Floor ${currentFloor} map`}
      className="map-image"
    />
    {filteredRooms.map((roomId) => (
      <div
        key={roomId}
        className="room-marker"
        style={{
          left: roomCoordinates[roomId]?.x,
```

```
            top: roomCoordinates[roomId]?.y,
            ...getRoomStyle(roomId),
           }}
           onClick={() => handleRoomClick(roomId)}
          >
           {roomId}
          </div>
         ))}
         {selectedRoom && (
          <div className="room-info">
           <div className="room-info-block">
            <h3 className="room-info-header">Кімната: {selectedRoom.id} </h3>
            <img
             src={`./image/controls/exitButton.png`}
             alt="Exit"
             className="room-info-header-image"
             onClick={() => setSelectedRoom(null)}
            />
           </div>
           <p>Кількість ліжок: {selectedRoom.beds}</p>
           <p>Вільні ліжка: {countFreeBeds(selectedRoom)}</p>
           <p className="room-info-container">
            Проживають:
            {selectedRoom.occupants && selectedRoom.occupants.length > 0 ? (
             <ul>
              {selectedRoom.occupants.map((occupant) => (
               <li
                key={occupant.id}
                style={{
                 ...getStudentStyle(occupant.violations),
                }}
                onClick={() => handleStudentClick(occupant.id)}
               >
                {occupant.name || "Невідомо"} - Акти:{" "}
                {occupant.violations ?? "0"}
               </li>
              ))}
             </ul>
            ) : (
             " Нікого"
            )}
           </p>
          </div>
         )}
        </div>
       </div>
      );
     };
     export default HomePage;
```

ДОДАТОК Е

Акт впровадження

УЗГОДЖЕНО
Декан факультету інформаційних технологій та комп'ютерної інженерії, к.пед.н., доцент
_____ Світлана КИРИЛАЩУК
«____» _____ 2025 р.

ЗАТВЕРДЖУЮ
Проректор з науково-педагогічної роботи та організації освітнього процесу ВНТУ, к.т.н., доцент
_____ Олександр ПЕТРОВ
«____» _____ 2025 р.

АКТ ВПРОВАДЖЕННЯ № ____

результатів бакалаврської кваліфікаційної роботи

Замовник Вінницький національний технічний університет
(найменування організації)
Цим актом підтверджується, що результати роботи – «Комп'ютерна онлайн система обліку, рейтингування та контролю проживання студентів у гуртожитку» що виконана студентом гр. 1КІ – 23мс, ВНТУ, Маліцьким В.В.
(виконавець)
впроваджено у Вінницькому національному технічному університеті
(найменування організації, де здійснювалося впровадження)
1.Вид впроваджених результатів вебзастосунок
(експлуатація виробу, роботи, технології)
2. Характеристика масштабу впровадження одиничне
(унікальне, одиничне, партія, масове, серійне)
3. Форма впровадження програмний продукт
4. Новизна результатів роботи розширення функціональних можливостей попередніх розробок
(піонерські, принципово нові, якісно нові, модифікації, модернізація старих розробок)
5. Впроваджені: в мережі ВНТУ
6. Річний економічний ефект ____-

Від виконавця:
Студент групи 1КІ-23мс
_____ Владислав МАЛІЦЬКИЙ

Від організації:
Завідувач кафедри обчислювальної техніки ВНТУ, д.т.н., професор
_____ Олексій АЗАРОВ

Науковий керівник
_____ доц. Олена ВОЙЦЕХОВСЬКА